Generación de historias centradas en el usuario con plataformas móviles

TRABAJO DE FIN DE GRADO

Christian Álvarez Sánchez
Kevin Darío Arboleda Arturo

Dirigido por

Gonzalo Méndez Pozo
Raquel Hervás Ballesteros

Departamento de Ingeniería del Software
e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Junio 2016

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

Generación de historias centradas en el usuario con plataformas móviles

Ingeniería del Software e Inteligencia Artificial

Dirigida por el Doctor

Gonzalo Méndez Pozo
Raquel Hervás Ballesteros

Departamento de Ingeniería del Software
e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Junio 2016

A todos aquellos que
alguna vez escribieron
código en un folio en blanco.

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

Christian Álvarez Sánchez

Kevin Darío Arboleda Arturo

Agradecimientos

En un agujero en el suelo, vivía un hobbit.

Inicio de El Hobbit. J.R.R.Tolkien

En primer lugar quería agradecer a Gonzalo Méndez Pozo y a Raquel Hervás Ballesteros, nuestros directores, por todo su apoyo durante la realización del proyecto. Su disponibilidad y apoyo moral sirvieron para mirar siempre hacia adelante y a esforzarme, dando las directrices necesarias para que mis ideas llegaran a buen puerto. Gracias a los compañeros, que han estado en los buenos y malos momentos. Tantas prácticas y tardes de biblioteca han servido para que no seamos solo compañeros, sino amigos. Gracias en especial a mi compañero en este proyecto, Kevin, que siempre hizo las largas tardes de estudio un poco más amenas. Gracias a la Universidad Complutense de Madrid y en concreto a la Facultad de Informática, por haberme dado la posibilidad de estudiar aquello que me gusta y me apasiona. Gracias a todos los profesores que, a lo largo de estos cuatro años, me han impartido clase. Cada uno de ellos ha aportado su granito de arena para que este proyecto pudiera ver la luz. Gracias a Ángela Brea Fernández que, además del apoyo artístico de la aplicación, siempre estuvo a mi lado en cada paso del camino. No podría tener mejor compañera de viaje. Eres la luz que me guía. A tu lado, mi carcajada es más sonora, mi sonrisa mas brillante y mi vida mejor. Gracias a mis abuelos, que me enseñaron el amor por la ingeniería y que el esfuerzo recibe su premio aunque sea a largo plazo. Gracias a mi hermano Jorge, por haber soportado a un hermano mayor siempre con una sonrisa. Y por último y sí, más importante, a mis padres Jesús y Nila. Gracias por todo el esfuerzo y la paciencia que habéis tenido conmigo. Por todas las oportunidades que me habéis brindado siempre con el mayor cariño del mundo. Vuestra influencia, guía y consejos me han hecho ser quien soy. Gracias papá. Gracias mamá. Gracias por mi vida.

-Christian Álvarez Sánchez-

Mis más sincero agradecimiento a Gonzalo Méndez Pozo y a Raquel Hervás Ballesteros, por habernos brindado la oportunidad de realizar este proyecto, ya que sin su guía y apoyo esto no podía haber salido adelante. Gracias por supuesto, a ese grupo de estudio que ha estado conmigo desde el principio del camino, porque sin todas las risas, los enfados y todo lo vivido yo no sería quien soy y estaría donde estoy ahora. Mil gracias a mis padres, por confiar y creer en mi en todo momento y enseñarme que con esfuerzo y constancia se puede conseguir lo que sea. Y agradecer sobre todo y en especial a mi compañero Christian, por estar ahí siempre, por aguantar con infinita paciencia cada una de mis tonterías, por enseñármelo todo y por ayudarme en cada paso del camino, ya que sin él, yo no habría llegado hasta aquí. Gracias amigo.

-Kevin Darío Arboleda Arturo-

Resumen

Dado que la textura del Universo es la más perfecta y la obra de un Creador sapientísimo, nada sucede en el Universo sin obedecer alguna regla de máximo o mínimo.

Leonhard Paul Euler

Dentro del área de la creatividad computacional, una de las aplicaciones que despiertan mayor interés actualmente es la de la generación de historias, con aplicaciones prácticas dentro del mundo empresarial (por ejemplo, en coaching). Habitualmente, esta generación no se realiza en tiempo real, debido al coste computacional que requiere el proceso de generación. En el presente proyecto se propone la construcción de un generador que funcione de manera similar al juego The Stanley Parable, donde las historias se generan, de una manera muy reducida, a medida que el jugador se mueve y explora el escenario del juego. Se propone, por tanto, la creación de un generador de historias basado en el uso de plataformas móviles, que cree en tiempo real una historia basada en los movimientos del jugador por un entorno real.

Índice

Índice de figuras

Capítulo 1

Introducción

No he perdido contra la dificultad, si no contra el tiempo.

Leonardo di ser Piero da Vinci

Resumen: Este capítulo presenta una breve introducción a PARABLE. Nos dará una visión general de lo que trata el proyecto y sus partes

1.1. Introducción

Desde el principio de los tiempos el hombre ha querido narrar su historia, así como dejar por escrito los conocimientos adquiridos con la finalidad que las siguientes generaciones se puedan beneficiar de ello y aprender. Prueba de ello es la cantidad de libros de distintas disciplinas que aportan conocimiento que tenemos hoy en día a nuestro alcance. Toda esta información contenida en la palabra escrita no se entendería sin una narrativa elaborada y comprensible. Por lo tanto podemos definir que la narración es el resultado de hablar, o contar lingüísticamente o visualmente una sucesión de hechos que en este caso se puede considerar historia.

Una narración siempre presenta varios elementos que deben confluir para poder ser considerada una narrativa completa. Debe contener, al menos, un actor, que es aquel elemento que experimenta los hechos o sucesos que ocurren y el otro elemento, por supuesto, son los hechos en cuestión que suceden. Una vez que estos dos elementos confluyen, hay que considerar que los hechos que le ocurren al actor estén relacionados, encadenados y que se vayan sucediendo de una forma más o menos lógica. Los textos narrativos se caracterizan por recrear el sentir del usuario, describir los lugares o espacios

donde ocurre la acción y el camino. Así pues, definimos una narración como los hechos que le suceden a un actor o usuario que transcurren en diferentes escenarios con inicio, nudo y desenlace común y relacionados lógicamente entre ellos.

Por último, dentro de la narración, no podemos olvidar la figura del narrador. Este elemento, que puede ser el propio actor o un narrador externo, nos relata la historia y nos describe las situaciones y lugares en los que transcurre la narración.

La Generación de Lenguaje Natural es un tipo de narración generada por ordenadores. Este tipo de creación de lenguaje conlleva varios desafíos: el léxico, ya que una misma palabra puede tener varios significados y este problema debe de ser resuelto para no provocar errores de comprensión; la estructura, ya que la morfología de una frase influye directamente en su significado y por último el nivel pragmático que puede dar lugar a que una frase no signifique lo que se está diciéndose como es el caso de la ironía.

Entendemos la generación de lenguaje natural como el proceso de construcción de texto en lenguaje humano para la comunicación con un fin específico. Para generar este texto, el sistema debe escoger cierta información de una base de conocimiento, decidir como organizarla, y determinar como mostrar y producir el texto adaptado al lenguaje, lo cual incluye decidir el léxico y las estructuras sintácticas que se van a utilizar. Este tipo de generación se puede observar en algunas aplicaciones como Stanley's Parable[1] que es un juego en el que el jugador se puede mover libremente y realizar acciones con elementos de su alrededor. Esta historia es presentada a través del narrador donde este sugiere un camino y el usuario puede optar por contradecir o realizar la sugerencia con lo que la narración debe adaptarse a las acciones del jugador. El objetivo principal del proyecto es la Generación de Lenguaje Natural con la finalidad de ofrecer al usuario una narrativa interesante, completa, lógica y rica que narre la experiencia del usuario dentro de un edificio.

Hoy en día casi todas las personas de nuestro entorno tienen dispositivos móviles que, además, usan a diario y los acompañan en todo momento. Empleamos los móviles en numerosas situaciones a lo largo del día, desde su uso normal como es una llamada telefónica, hasta el acceso a información en internet o el camino más corto a nuestro destino. Utilizando un GPS, podemos conocer nuestra posición en exteriores con una precisión aceptable. Sin embargo, la precisión dentro de los edificios debe ser mucho más exacta ya que una desviación de unos pocos metros puede hacer que estés en una habitación o en otra. Esta precisión no puede ser conseguida por el GPS así que aprovechando la omnipresencia de los móviles se nos planteó la pregunta

[1] http://www.stanleyparable.com/

de si podríamos emplear alguna tecnología disponible en la mayoría de los móviles para determinar la posición de un usuario a partir de su dispositivo y poder narrar las circunstancias en las que se encontrase.

Aplicando los conceptos antes descritos presentamos PARABLE. Se trata de un proyecto que intenta demostrar las posibilidades que ofrece la unión de la Generación de Lenguaje Natural y el posicionamiento WiFi dentro de edificios que, en nuestro caso, usamos como escenario la Facultad de Informática. Al ejecutar la aplicación, el usuario tendrá como objetivo alcanzar un determinado punto de la facultad. Para ello nos vimos en la necesidad de conocer, en primera instancia, la situación del usuario de manera suficientemente precisa como para distinguir entre aulas. Para esta necesidad, utilizamos como base el algoritmo desarrollado por un proyecto de años anteriores llamado AVANTI. A continuación, dividimos la facultad por pasillos, secciones y aulas dando a algunos de ellos la característica de punto de interés y elaboramos un sistema de coordenadas que referenciaría la posición del usuario a lo largo del edificio.

Una vez que la aplicación nos ha localizado y nos ha dado nuestro objetivo podremos empezar con el recorrido. Durante el trayecto, la aplicación, mediante la Generación de Lenguaje Natural, nos va narrando nuestro paso por la facultad así como los diferentes *checkpoints* que tendremos que alcanzar. Además, durante el recorrido habrá situados unos determinados códigos QR que añadirán y enriquecerán el texto una vez que sean encontrado. Estos códigos también constan de una serie de puntuaciones que deberemos recolectar para poder ganar la aventura en un futuro. Durante todo el proceso podremos hacer caso a las ordenes de PARABLE o desobeceder, podremos entrar en zonas fuera de cobertura, o incluso permanecer mucho tiempo en un mismo lugar. Todo ello se verá reflejado en la narración mediante frases construidas según la situación. Para finalizar, estas frases estarán enriquecidas usando varios sinónimos tanto para verbos, adjetivos o sustantivos con el fin de no dar sensación de repetición.

Es de sobra conocido que las redes sociales tienen un gran peso en nuestra sociedad. Muchas personas, hoy en día, están en este tipo de plataformas que ofrecen al usuario la posibilidad de compartir sus vivencias con el resto del mundo. Aprovechando esta situación quisimos que PARABLE proporcionara varias de estas opciones al usuario para que pudieran compartir su experiencia. PARABLE nos ofrece, durante el uso de la aplicación, la posibilidad de poder *tweetear* nuestra aventura. Además de esta funcionalidad, contaremos con la posibilidad de poder enviar nuestra historia a través de mail para poder guardarla y poder revivirla en un futuro. También tendremos la opción de poder enviarla a la página web de PARABLE para que así haya otros usuarios que la puedan leer. Por último, PARABLE, también consta

de tecnología *text to speech*, la cual, mediante audio, nos narra la historia.

En conclusión, utilizando el posicionamiento por WiFi y la Generación de Lenguaje Natural, queremos comprobar y demostrar a lo largo del presente proyecto que es posible realizar una actividad similar al juego Stanley's Parable con la diferencia de que en el nuestro, se pueda realizar en entornos reales y en el interior de edificios.

1.2. Prologue

From the beginning of time men have wanted to tell their story and to write knowledge acquired amount so that future generations benefit from it and learn. Proof of this is the large of books from different disciplines whose content provides knowledge. All the information that the written word comprises could not be understood without an elaborate and comprehensible narrative. Therefore we can define narrative as the result of telling a succession of events in a way that they can be considered story. A story always includes several elements that must be put together in order to be considered a complete narrative. On the one hand it must contain at least one actor, who is the element suffering facts or events occurring. On the other hand, of course, it needs relevant facts to happen. Once these two elements are together, it is also necessary for these events to be related and chained in a logical succession. The narratives are characterized by recreating the feelings of user, describing the places where the action takes place. Thus, we define a narrative as the events that happen to an actor and that take place in different scenarios with beginning, middle and common outcome, always logically related to each other.Finally, in narrative, we can not forget narrators. This element, which can be the actor itself or an external narrator, tells the story and describes situations and places where the story occurs.
Natural Language Generation is a type of narrative created by computers. This type of language involves several challenges: lexicon, because an unique word can have several meanings and this problem must be resolved to avoid misunderstandings; the structure, due to the fact that the morphology of a sentence influence their meaning; and finally, the pragmatic level, which implies that a sentence does not mean what is being literally said, as in the case of irony. We understand Natural Language Generation as the process of building text in human language to communicate with an specific purpose. To generate this text, the system must first select information from a knowledge database, then decides how to organize it, and finally determines how to produce text display and to adapt it to the natural language. This step includes deciding lexical and syntactic structures that are to be used. This type of generation can be seen in some applications such as Stanley's

Parable [2] which is a game then where players can move freely and perform actions with elements around. The story is presented through the narrator, who suggests a path the user can choose or decline. This is to say that the narrative must adapt to player's actions. The main objective of this project is to use Natural Language Generation in order to offer the user an interesting, complete, logical and rich narrative, set in the interior of a building.
Nowadays everyone owns mobile devices, and keeps them close at hand. For instance, we use mobile phones in many situations throughout the day: from its normal use which is making phone calls, to other more complex functions such as surfing on the Internet or finding the easiest and shortest way to get to our destination. By using GPS, we can know our position outdoors with acceptable accuracy. However, this accuracy within buildings should be much more precise since a deviation of a few meters can can wrongly determine that we are in one room or another. This accuracy can not be achieved by the GPS so by taking advantage of the ubiquity of mobile, we raised the question of whether we could use some technology available in most phones to determine the position of an user from his device and ALSO be able to tell the circumstances where it was found.

Applying concepts described above, we present PARABLE. This is a project that tries to show the possibilities offered by the connection between the Natural Language Generation and WiFi positioning within buildings. In our case, we use as scenario the building in Facultad de Informática at the Complutense University of Madrid. When running our application, the user will have as an objective to reach a certain point of the faculty. For this purpose, first we needed to know the user's situation with accuracy enough to distinguish between classrooms. To this end, we used as a basis the algorithm developed by a project called AVANTI on previous years. Then we divided the faculty into hallways, classrooms and sections marking some of them as relevant for our application and drew up a coordinate system that would reference user's position throughout the building.

Once we have been located by the application and it has given us our goal, we begin our tour. During our trip, PARABLE, by using Natural Language Generation, narrates our passage throughout the building and also the different *checkpoints* we have to achieve. In addition, during the tour there will be situated some QR codes that enrich the text when found. These codes also consist of a series of scores that we must collect to win the adventure in a future.

Throughout the process we can heed or disobey PARABLE's orders, we have the possibility to enter unreachable areas or even to stay long in the same place. All these decisions will be reflected in the narrative with phrases built according to the situation. We also have to bear in mind that all these

[2] http://www.stanleyparable.com/

phrases are enriched using several synonyms for verbs, adjectives and nouns, in order not to give a sense of repetition.
It is well known that social networks have great weight in our society. Many people nowadays are in these kind of platforms which offer the option to share experiences with the rest of the world. Noticing this situation, PARABLE wants to offer several of these options to the users so they could share their experiences. PARABLE offers, while using the application, the possibility of *tweeting* their adventure. In addition to this functionality, we have the possibility to send our story through an e-mail to save it and read it in the future. We also have the option to send it to PARABLE's website so that other users can read it. Finally, PARABLE, has also *text to speech* technology, which is capable to tell us the story by audio.

In conclusion, using WiFi positioning and Natural Language Generation, we want to ensure and demonstrate throughout this project that it is possible to perform a similar activity to the game Stanley's Parable with the difference that ours can be performed in real environments and inside of buildings.

Capítulo 2

Estado del Arte

La perfección es inalcanzable, pero si perseguimos la perfección, podremos alcanzar la excelencia.

Vince Lombardi

Resumen: Este capítulo da a conocer las últimas investigaciones y las líneas actuales de acercamiento a los temas en los que se centra el proyecto: Programación ubicua, Posicionamiento y Procesamiento de Lenguaje Natural.

2.1. Programación ubicua

La Programación Ubicua (Wikipedia, a) es entendida como la integración de elementos informáticos en el entorno de las personas. El concepto es atribuido a Mark Weiser (Weiser) quien lo definió de esta manera en sus artículos en 1988 (Weiser, 1988) cuando trabajaba para Xerox en el laboratorio de Palo Alto (PARC). Weiser escribió un artículo en 1991 (Weiser, 1991a) dando lugar a algunos de los primeros trabajos sobre el tema, en gran parte para definirlo y esbozar sus principales preocupaciones (Mark Weiser, 1999) (Weiser, 1996). Desde hace unos años también se denomina como Inteligencia Ambiental. Esta disciplina propone la creación de entornos físicos inteligentes que se adapten a las necesidades, gustos e intereses de la gente que vive en ellos, ayudando a llevar a cabo tareas diarias mediante la integración informática en el entorno de la persona, de forma que los ordenadores no se perciban como objetos diferenciados. En esta línea se investiga y desarrollan métodos de Ingeniería de Software para dotar de disciplina y rigor al desarrollo de sistemas basados en programación ubicua.

De este modo, la Inteligencia Ambiental se asienta sobre cuatro fundamentos principales:

Uso de espacios eficaces: Se basa en la detección del estado del usuario en el lugar en el que se encuentre y de sus necesidades. Este espacio se convierte en eficaz cuando intervienen, colaborando, varios dispositivos para dar mejor soporte y ayuda al individuo.

Invisibilidad: Este fundamento dictamina que los dispositivos deben desaparecer de la conciencia del usuario y que realicen sus operaciones más allá del uso directo. En este aspecto hay varias líneas de investigación como pueden ser el reconocimiento de texto (ABBYY[1]), voz (Dragon[2]), gestos (Kinect[3]), comprensión del lenguaje natural (Siri[4]). Aunque aún se está lejos de conseguir que los dispositivos desaparezcan completamente de nuestra consciencia, la tendencia de los dispositivos cotidianos sigue este fundamento. Esto lo podemos comprobar, por ejemplo, en los SmartWatch que toman medidas biométricas o geolocalización y reaccionan ante ellas.

Escalabilidad local: El concepto de localidad de servicios en computación ubicua es fundamental frente a la universalidad de servicios de Internet. Los usuarios realizan acciones dependiendo del contexto en el que se encuentren. Es decir, la computación ubicua debe conocer la situación del usuario para aplicar las acciones correctas. Por ejemplo, no tendría sentido que una aplicación de uso exclusivo en la oficina estuviera en uso si el usuario no se encuentra en ella.

Ocultación de los desniveles de acondicionamiento: La infraestructura y el desarrollo tecnológico disponibles intervienen directamente en los servicios ofrecidos y hoy en día esos servicios son poco uniformes. Esto entra en conflicto con el fundamento de invisibilidad ya que los sistemas que incorporan Programación Ubicua están aislados y no tienen una continuidad entre ellos.

Así pues, este tipo de computación debe ser proactiva, entendiendo como proactividad, que los dispositivos dejen atrás su comportamiento automático, y pasen a tener un comportamiento activo, con consciencia del entorno en el que se encuentran, en definitiva, un comportamiento inteligente (Weiser, 1991b). Estos dispositivos, a su vez, deben ser sensibles a los cambios en la información del entorno, como puede ser la localización y estado de los usuarios y otros dispositivos (Saha, 2003). Otro aspecto es la identificación de la información, recibida, en este caso, desde todas las fuentes posibles

[1] https://www.abbyy.com/textgrabber/
[2] http://www.nuance.es/dragon/index.htm
[3] http://www.xbox.com/es-ES/xbox-one/accessories/kinect-for-xbox-one
[4] https://es.wikipedia.org/wiki/Siri

aprovechando tanto la información dada por el usuario, como la información recogida por los sensores del dispositivo como podrían ser la inclinación, velocidad, geolocalización o incluso datos biométricos. Por último, como inteligentes que son, estos sistemas deben conocer las necesidades del usuario en cada momento para que éste se centre en lo que tiene que hacer y no en las herramientas que le permiten conseguir el fin Mark Weiser ha propuesto(Weiser, 1991a) tres modelos básicos que pueden ser considerados para desarrollar sistemas ubicuos que tienen como relación el ser planos, grandes y que tienen una salida visual:

Tabs: dispositivos de escasos centrímetros, que pueden ser llevados por un usuario.

Pads: dispositivos del tamaño de una mano.

Boards: dispositivos que pueden llegar a medir metros.

Si se relajan estas consideraciones (tamaño, respuesta visual, etc) entonces se puede extender este rango hasta un número de dispositivos mucho mayor, y también hasta un número de dispositivos mucho más útiles. Por lo tanto, se han acabado proponiendo estas tres clasificaciones:

Dust: dispositivos miniaturizados que pueden no tener algún tipo de salida visual MEMS (Sistemas Microelectromecánicos) incluir acrónimo, cuyo tamaño puede oscilar entre nanómetros hasta milímetros.

Skin: Pueden ser fabricados con capacidades de emitir luz y con diversos materiales, como polímeros conductivos, algunos dispositivos orgánicos. Se ven frecuentemente como ropa, elementos de decoración, etc.

Clay: Conjuntos de distintos MEMS que pueden combinarse para crear formas en tres dimensiones.

En definitiva, la Inteligencia Ambiental tiene como objetivo que el usuario realice sus tareas mientras los dispositivos electrónicos actúan "por detrás", recogiendo información, tratándola y respondiendo ante los estímulos (dados por los sensores y por el usuario) del medio en el que se encuentran sin que el usuario participe activamente en el uso de los propios dispositivos. Actualmente, la computación ubicua gira en torno a un amplio abanico de áreas de investigación: adquisición de señales corporales, RFIDs, Internet como soporte para computación ubicua, seguridad, sistemas Peer-to-Peer, sistemas de posicionamiento y localización, monitoreo y un largo etc. El futuro de la Programación Ubicua nos indica que nos estamos moviendo de un sistema casi descentralizado, compuesto por micro-ordenadores y *mainframes* hacia un sistema completo de programación ubicua. Como se pone de manifiesto en el proyecto de New Songdo City. Una ciudad ubicua (o ciudad-U) que se

está construyendo en una isla frente a la ciudad de Inchon, a 60 kilómetros al oeste de Seúl (Corea del Sur). Todos los sistemas de información estarán interconectados y las computadoras estarán integradas a las viviendas, las calles y los edificios de oficinas.

2.2. Posicionamiento y localización

El posicionamiento de los terminales móviles cada vez toma una mayor importancia con el aumento de su uso. Además muchas aplicaciones requieren una información precisa sobre la localización del propio terminal. Podemos conseguir la localización tanto a través de GPS o GSM, si nos encontramos al aire libre. Si nos encontramos en interiores, debido a que las otras redes no tienen la suficiente precisión dentro de los edificios, y teniendo en cuenta que cada vez más edificios cuentan con puntos de acceso inalámbricos y con más intensidad, se abre una nueva vía para el posicionamiento *indoor* que se consigue mediante las redes WiFi.

2.2.1. Posicionamiento con GPS

GPS (del inglés *Global Positioning System*) es un sistema de navegación por satélite que proporciona información precisa sobre la localización de un terminal, y está preparado para otra serie de usos como información meteorológica. Resumiendo bastante la parte técnica, puede ser usado en cualquier lugar donde hay línea directa y sin obstáculos hacia por lo menos cuatro satélites de la red GPS. El GPS fue originalmente desarrollado por el Departamento Americano de Defensa, e inicialmente basaba su funcionamiento en la señal recibida de 24 satélites. Se empezó a utilizar en 1973 para hacer frente a todos los problemas de la navegación de la época. Algunos países permiten el uso de repetidores GPS para permitir la recepción de señales GPS en interiores, o en localizaciones donde la cobertura no llega como se desearía. No obstante, las leyes de la Unión Europea y del Reino Unido prohíben explícitamente el uso de estos dispositivos, ya que las señales pueden causar interferencias en otros dispositivos.

Dados los obstáculos que el GPS encaraba a la hora de realizar el sistema de posicionamiento para PARABLE, fue rápidamente descartado en favor del posicionamiento mediante WiFi que nos permite posicionamiento en interiores sin el problema de las interferencias.

2.2.2. Posicionamiento con GSM

Es un servicio ofrecido por las compañías de telefonía móvil que permite determinar, con cierta precisión, donde se encuentra físicamente un terminal móvil determinado. La precisión deja mucho que desear si hablamos de

situaciones de interior ya que se encuentra entre los 200 metros y 2 kilómetros, y de 3 a 4 kilómetros en espacios rurales. Uno de los métodos que se emplean, es ID de célula. Este método es el cálculo de la posición teniendo en cuenta la Célula de origen y de las intensidades de las células adyacentes y sus frecuencias. Como podemos observar en la Figura 2.1, las estaciones base ofrecen cobertura en determinadas áreas a una determinada intensidad y según se va moviendo el dispositivo, las intensidades recibidas desde las estaciones base cambian, así como también puede cambiar la célula en la que nos encontramos. En la Figura 2.2 se toma como ejemplo un factor de reutilización 7 para que las frecuencias no se solapen. Este método tiene una mejora llamado ID de célula mejora que consigue mejor precisión en zonas rurales pero que sigue siendo impreciso para lo que nuestro proyecto necesita.

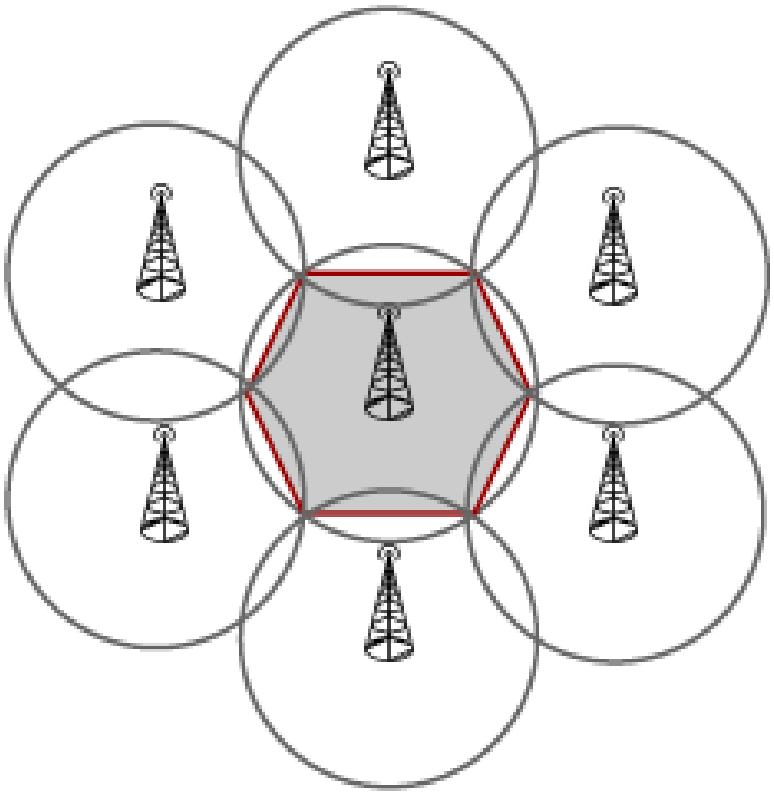


Figura 2.1: Representación de estaciones base para el posicionamiento GSM.

2.2.3. Posicionamiento mediante Beacons

Consiste en una red de sensores que envían información de los usuarios. Estos recogen esa información y con algoritmos más o menos complejos, el

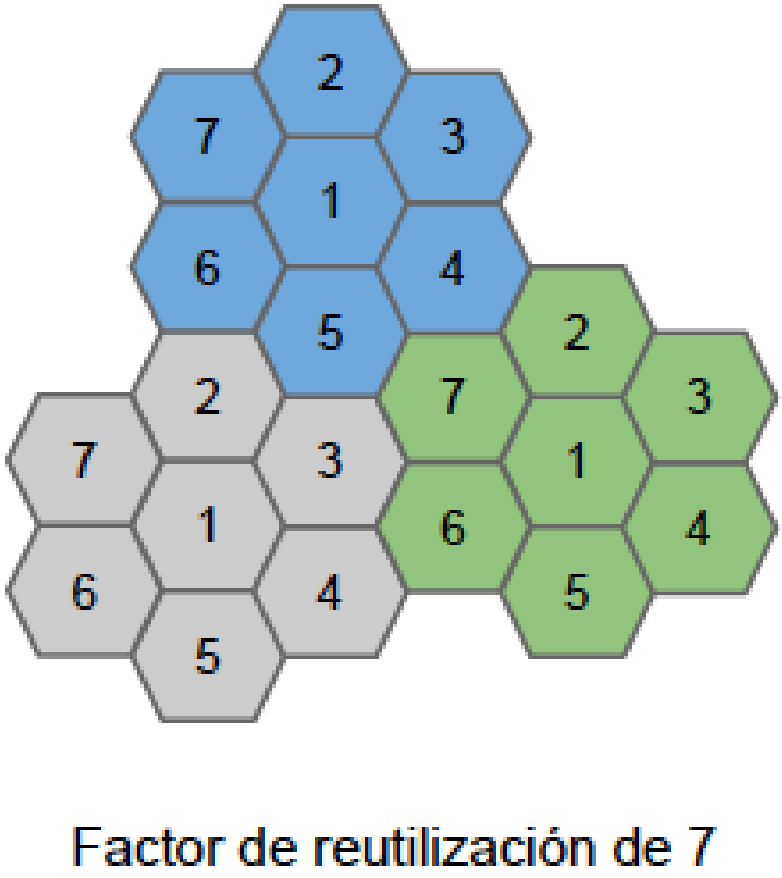


Figura 2.2: Representación de las Celdas para el posicionamiento GSM.

sistema es capaz de posicionar a esos usuarios en algún punto de esa zona de cobertura (Research, 2004) y (Nogee, 2001). Los beacons son dispositivos de bajo consumo, suficientemente pequeños como para fijarse en paredes. Utilizan una conexión de bajo consumo denominada *Bluetooth Low Energy* (BLE) para transimitir información sin necesidad de una sincronización de los aparatos. Esta señal es captada por los dispositivos y enviada a un servidor en la nube a través de internet. Este procesa la información, la analiza y da como resultado una respuesta (Wikipedia, 2016). El hardware de estos dispositivos es bastante simple. Consiste en un microcontrolador con un chip de radio bluetooth y una batería. Esto da como resultado que su valor de mercado muy asequible (Bekkelien, 2012).

Las aplicaciones son extensas. Como es un sistema de posicionamiento, utiliza técnicas de triangulación basadas en la información proporcionada por la potencia de la señal recibida, los beacons o balizas, tienen una localización

fija en el entorno y su posición es conocida. A partir de eso y con un sistema de coordenadas prefijado es posible calcular la posición del usuario dentro de un entorno. Otra de las aplicaciones es la publicidad. Son herramientas de marketing o *geomarketing* que pueden predecir la situación del cliente en el entorno y con ayuda de aplicaciones comerciales ofrecerle productos y servicios. Otra de las aplicaciones es la localización de objetos. Estos objetos enviarían datos periódicamente a un servidor de su posición y estado y se conseguiría una mayor eficiencia en su tratamiento.

2.2.4. Posicionamiento mediante RFID

Otra de las formas de conseguir un posicionamiento adecuado indoor es el proporcionado por las etiquetas RFID (Radio Frequency IDentification). Un sistema RFID consta de los siguientes tres componentes:

- Etiqueta RFID: compuesta por una antena, un transductor radio y un chip. La misión de la antena es proporcionar al chip la capacidad de poder transmitir la información que contiene. Hay varios tipos de etiquetas, solo lectura con id único, lectura y escritura donde la información puede ser modificada y anticolisión que permiten que un lector identifique varias al mismo tiempo.

- Lector de RFID: compuesto por una antena, un transceptor y un decodificador. El lector envía señales periódicamente para comprobar si hay alguna etiqueta cerca de él y cuando capta una señal, recupera la información y la envía al sistema que procesa los datos.

- Sistema de procesamiento de datos: proporciona los medios de proceso y almacenamiento de datos.

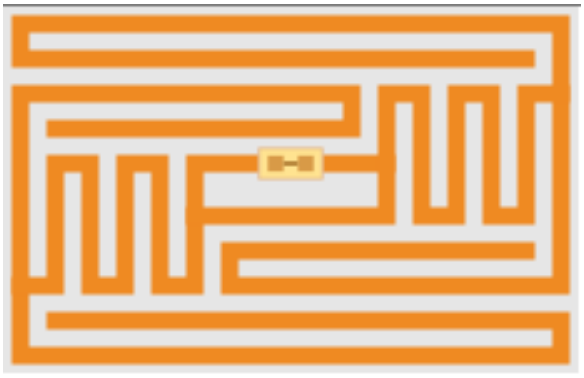


Figura 2.3: Etiqueta RFID.

Hay tres tipos de etiquetas:

- Pasivas: aquellas que no poseen alimentación eléctrica. La señal recibida induce una pequeña corriente suficiente para operar.
- Activas: poseen su propia fuente de alimentación y son mucho más fiables
- Semipasivas: poseen fuente de alimentación propia pero en lugar de alimentar la transmisión de la señal, lo que alimentan es el microchip.

Las RFID tienen multitud de aplicaciones, desde la identificación de objetos hasta eventos y mercadotecnia pero nosotros nos centraremos en el uso que se le puede dar como elemento de posicionamiento. El posicionamiento mediante RFID se basa en la colocación en el entorno de etiquetas con información sobre el mismo entorno, es decir, la localización exacta de la etiqueta e información adicional. Cuando el usuario se sitúe cerca de la etiqueta, su dispositivo la identificará y puede enviar su posición exacta para el procesamiento de datos además de recibir datos de la propia etiqueta. La información enviada por la etiqueta puede proporcionar idea de los movimientos que efectúan los usuarios dentro de un centro comercial, cuales son los productos más atractivos y así gestionar mejor el tiempo y el espacio. Otra aplicación es la relacionada con las ciudades inteligentes. En ellas se colocan etiquetas RFID que dan información sobre el estado del tráfico, reconocimiento de señales por parte del vehículo, la cantidad de aparcamiento disponible en una zona, etc. Por último, también pueden ser usadas en logística ya que permiten tener localizado cualquier producto dentro de la cadena de suministro (Wikipedia, 2015).

2.2.5. Posicionamiento con WiFi

Cuando nos encontramos en el interior de un edificio, las condiciones para el posicionamiento por GPS se endurecen y se hace más difícil la localización. Esto, unido al crecimiento enorme de puntos de acceso para redes WiFi tanto en ambientes domésticos como en oficinas o centros comerciales, permite utilizar alternativas al GPS como es el posicionamiento WiFi. Ciertas técnicas ya han sido desarrolladas para la localización de móviles a partir de la integridad o intensidad de la señal WLAN(*Received Signal Strength*, o RSS en inglés). Es el caso del sistema RADAR (Bahl y Padmanabhan, 2000) que utiliza mapas precalculados de potencias de señal para lograr su objetivo. Como podemos ver en la Figura 2.4, en un determinado punto confluirán tres o más redes WiFi y a partir de sus intensidades previamente calculadas podremos determinar nuestra posición dentro del edificio.

El posicionamiento basado en WiFi (*WiFi Positioning System*, o WPS en inglés) es un sistema que también se puede aplicar en áreas urbanas gracias a que cada vez más ciudades proveen de redes en espacios públicos o sitios de interés. SkyHook Wireless es una compañía puntera en este tipo de servicio

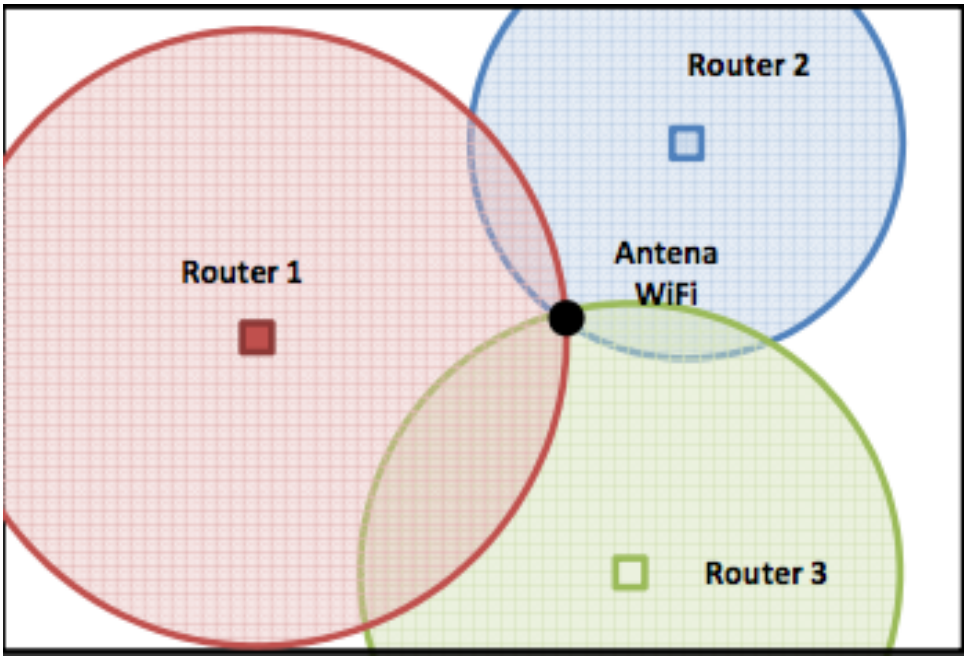


Figura 2.4: Triangulación de posición mediante las intensidades de las redes WiFi.

que contiene una base de datos pública a la que se puede acceder a través de su API y obtener el posicionamiento basado en los puntos de acceso accesibles desde un terminal.

2.3. Procesamiento de Lenguaje Natural

El Procesamiento de Lenguaje Natural (*Natural Language Processing*, o NLP en inglés) (Wikipedia, c) es un campo de las Ciencias de la Computación, Inteligencia Artificial y la Lingüística que estudia las interacciones entre las computadoras y el lenguaje humano (Gelbukh y Bolshakov, 2000). El NLP nace en la década de los 60 como subárea de la Inteligencia Artificial, con el fin de estudiar los problemas derivados de la generación y comprensión automática del lenguaje natural. Entre las aplicaciones principales del procesamiento de lenguaje natural se pueden mencionar tres:

- Recuperación de información. La aplicación del procesamiento de lenguaje natural más obvia y quizá más importante en el momento actual es la búsqueda de información (se llama también recuperación de información). Las técnicas más usadas actualmente para la recuperación de información involucran la búsqueda por palabras clave.

- Interfaces en lenguaje natural. Las computadoras están entrando en

todos los aspectos de nuestra vida cotidiana: en las oficinas, en las tiendas, en las escuelas, en los servicios públicos. Sin embargo, la gran mayoría de la gente no tiene la preparación adecuada para usarlas. Así que la máquina debe adaptar su lenguaje al de los humanos y comprender exactamente lo que el usuario quiere conseguir. Esto es de vital importancia y donde radica el problema principal, ya que si el dispositivo no comprende una orden puede dar lugar a situaciones de riesgo o consecuencias graves.

- Traducción automática. El esquema general de prácticamente cualquier traductor automático es el siguiente. El texto en el lenguaje fuente se transforma a una representación intermedia. De ser necesario, se hacen algunos cambios a esta representación. Después, esta representación intermedia se transforma al texto en el lenguaje final. Adicionalmente a los problemas de análisis de texto, la traducción automática enfrenta los problemas específicos para la generación de texto. Uno de estos problemas es la selección de palabras. Por ejemplo, para traducir del inglés la frase 'John pays attention to Mary', pay en este contexto no significa pagar con lo que esta variante de traducción es incorrecta: Juan paga atención a María. La manera correcta de representar la palabra pay es tratarla como una llamada función léxica: esta palabra significa la ejecución de la acción de atención por el agente (Juan). En el español, la palabra que indica la ejecución de atención es 'prestar'.

El procesamiento de lenguaje natural tiene multitud de aplicaciones, destacando entre ellas: Síntesis del discurso, análisis y compresión del lenguaje, reconocimiento del habla, etc.

Un claro ejemplo de que el NLP está muy de actualidad es el uso de asistentes de voz en los dispositivos móviles como son Siri, Cortana o Google Now. En el caso de Siri, aplicación de iOS (Wikipedia, d), su ultima versión es capaz de responder preguntas, hacer recomendaciones y realizar acciones mediante la delegación de solicitudes hacia un conjunto de servicios web que ha ido aumentando con el tiempo. Está perfectamente integrado en el ecosistema de iOS y ofrece interacción conversacional con otras aplicaciones, tales como los recordatorios, consulta del estado del tiempo, la bolsa, el servicio de mensajería, el correo electrónico, calendario, contactos, notas, música, reloj, navegador web, Wolfram Alpha y los mapas.

Otro ejemplo, aplicado en este caso, es el juego Stanley Parable (Galactic, 2011). Este juego nos va dando órdenes que podemos cumplir o no, y ante nuestras elecciones reconstruye la historia y realiza diferentes interacciones con el usuario. Esto hace que la experiencia con el juego sea más plena y el jugador se sienta más integrado en la historia del videojuego.

El NLP se ha empezado a aplicar también en el campo de la Lectura Fácil. El proyecto Simplext (Simplext), liderado por Technosite, una empresa

de la Fundación ONCE, en consorcio con varias empresas y universidades, ha hecho algunos desarrollos en esa línea. Hasta la fecha, han conseguido el desarrollo de un algoritmo que permite una alineación entre un texto original y un texto adaptado. La finalidad era conseguir un adaptador automático centrado en noticias. La razón de la delimitación del tipo de texto se debe a la necesidad de crear glosarios específicos.

En esta misma línea, la Universidad de Alicante también está realizando investigaciones dentro del Grupo de Procesamiento del Lenguaje y Sistemas de Información. Recientemente, este grupo ha anunciado el desarrollo de una aplicación para favorecer la comprensión lectora en pantalla con apoyos para comprender el significado del texto, sin hacer, de momento, una adaptación específica como plantea la metodología de lectura fácil.

Las perspectivas del NLP para un futuro se están centrando en el desarrollo dinámico de la sintaxis y la pragmática del texto así como de la semántica, es decir, conseguir que la máquina entienda el mensaje subyacente que envía el usuario además del significado literal de la frase.

En PARABLE aplicamos un procesamiento del lenguaje natural mediante la narración de las situaciones que el usuario realiza mientras está en uso la aplicación para que el uso de la aplicación sea más dinámico y la experiencia de uso sea más completa.

2.4. Motivación y Objetivo

El procesamiento de lenguaje natural nos permite hacer que el usuario se sienta parte y partícipe del funcionamiento de una aplicación. Conseguir una experiencia más completa enfocada al propio usuario es el fin de todas las aplicaciones de entretenimiento como, por ejemplo, videojuegos que narren los pasos del jugador. Una de las principales causas de abandono de un juego es no sentirnos identificados dentro del mismo o que no acabemos de conseguir empatía por el personaje. Sin embargo, si es el propio usuario el protagonista del juego este problema disminuye ya que se toma todo de una forma más personal. Por otro lado, el guión o narrativa de los juegos no suelen tener en cuenta al usuario ya que la historia se centra en el personaje principal, es decir, jamás se rompe la cuarta pared (Wikipedia, b), (el juego no se dirige directamente al jugador). En el mercado no existen demasiadas aplicaciones que describan, en lenguaje natural, las actividades que el usuario está realizando. Una de esas aplicaciones que rompen con la norma y por ello la hemos tomado como inspiración es el juego Stanley Parable (Galactic, 2011). El juego es presentado al jugador desde una perspectiva en primera persona. El jugador se puede mover libremente y realizar acciones con ciertos elementos de su alrededor, como presionar botones o abrir puertas, pero no tiene otros controles. La historia es presentada al jugador principalmente a través de la voz del narrador del juego, que explica que el protagonista

Stanley trabaja en un edificio de oficinas, y tiene la tarea de monitorizar datos en una pantalla de ordenador y presionar botones según se requiera sin cuestionar nada. En este momento, la historia se divide en numerosas posibilidades, basadas en las elecciones del jugador. El narrador continúa la historia, pero cuando el jugador llega a un área en la que una elección es posible, este sugiere a Stanley qué ruta tomar. El jugador puede optar por contradecir al narrador y realizar otra acción, obligando a la narración a que se adapte a esta nueva dirección.

2.5. Descripción de PARABLE

El desarrollo de PARABLE ha querido aunar en un solo proyecto el posicionamiento WiFi y el procesamiento de lenguaje natural. Fusionando todas estas tecnologías, PARABLE, quiere conseguir una experiencia más completa en lo que a juegos se refiere. Buscamos que, con la generación de historias, el usuario se sienta dentro de la aventura a través de su terminal móvil y la experiencia sea más envolvente.

El posicionamiento en interiores fue un gran desafío debido a que el GPS no funciona adecuadamente dentro de edificios y tuvimos que valorar otras alternativas. Así que optamos por desarrollar un posicionamiento basado en las intensidades WiFi. Por ello, con el uso de terminales móviles y sus receptores WiFi y sensores conseguimos localizar al usuario y además, nos permite movernos dentro del campo de juego con total libertad, en este caso, la facultad. Recopilando toda la información recogida del móvil y del posicionamiento se creará una historia narrativa en torno al camino elegido por el jugador que será más placentera debido a que él es el protagonista.

PARABLE sigue el paradigma de la arquitectura de cliente-servidor. Una vez que el jugador ejecuta la aplicación, ésta establece una conexión con el servidor y todos los datos necesarios, a petición de la aplicación, son recogidos desde un servidor externo. Estos datos serán, tanto las intensidades WiFi en cada punto, como la información que se debe mostrar gracias a los códigos QR, así como las plantillas narrativas que también estarán alojadas en el servidor.

Capítulo 3

Desarrollo de PARABLE

Si he logrado ver más lejos, ha sido
porque he subido a hombros de gigantes.

Isaac Newton

Resumen: Este capítulo se presentarán las características de PARABLE en profundidad. Se explicarán las partes de las que consta el proyecto, las dificultades, las soluciones propuestas y las funcionalidades del mismo.

3.1. Introducción

PARABLE es un proyecto que se centra en la generación de lenguaje natural. Nuestro principal objetivo era conseguir que la aplicación narrara una historia de nuestro camino utilizando como medio los dispositivos móviles.

Esta desarrollado sobre Android5.0 (2014) y como entorno de programación usamos el oficial, Android Studio[1] y la versión es la 5.0. En un principio tuvimos problemas con esta versión ya que al ser, en su momento, la más actual hasta la fecha, los emuladores no funcionaban todo lo fluido que se esperaba de ellos. Además, el tener varios hilos de ejecución la aplicación sobre emulador no era adecuada para hacer pruebas y tests. Para hacer pruebas y probar la aplicación se nos prestó un Samsung Galaxy Note 3 que podía ejecutar perfectamente la versión de Android 5.0. Se valoró en un principio la posibilidad de realizarla en iOS[2] pero se desechó por los siguientes motivos:

- La licencia, a diferencia de Android no es gratuita y supone un desembolso importante.

[1] http://developer.android.com/intl/es/tools/studio/index.html
[2] http://www.apple.com/es/ios/

- Menor información de la funcionalidad interna y opacidad del sistema operativo.
- Comunidad de desarrollo de menor tamaño.
- No es código abierto frente a Android que sí lo es.

Por todas estas razones, PARABLE se desarrollo para dispositivos Android.

Además, hemos utilizado los siguientes lenguajes para obtener las funcionalidades que más se adaptaban a nuestras necesidades:

- PHP, para comunicarnos con la base de datos.
- HTML, CSS y jQuery, para dar soporte a la aplicación web.
- XML, para realizar la distribución de la facultad y etiquetar cada sección.

La aplicación, en primer lugar, nos posiciona dentro del edificio. Una vez encontrados nos da un objetivo, el cual sería llegar a un punto de la facultad pasando por diferentes *checkpoints*. Si nuestro camino es el indicado por la aplicación, la historia seguirá su curso normal, poniendo de manifiesto lo que estamos haciendo. Si, por el contrario, tomamos otro camino, la aplicación PARABLE también nos lo mostrará e intentará que retomemos el buen curso. Por supuesto, en cualquier momento, se puede retomar el camino correcto o desobedecer.

Los *checkpoints* por los que tenemos que pasar estarán indicados tanto por un código QR que tendremos que validar y reconocer, o por entrar simplemente en el area que contiene el el *checkpoint*. Esta funcionalidad se realizó pensando, en el futuro, en la aplicación de PARABLE como pudiera ser el Código Santesmases[3] que se trata de una juego elaborado por la Facultad de Informática de tipo *gymkana*. Además, el usuario de la aplicación deberá buscar, a lo largo del camino, códigos QR que contendrán diferentes puntuaciones dependiendo de la dificultad y que también enriquecerán la historia.

Las Figuras 3.1 dan una muestra de lo que el usuario ve cuando accede a la aplicación. En primer lugar una portada y a continuación, el menú principal desde el que se gestiona la aplicación. En este menú, tenemos:

- Comenzar relato, que nos da acceso a la funcionalidad principal donde se mostrará la narrativa.
- Guardar coordenadas, donde se pueden añadir las coordenadas de nuevas secciones a la base de datos.

[3] https://www.youtube.com/watch?v=P4jsIweZvig

(a) Portada de PARABLE (b) Menú principal de PARABLE

Figura 3.1: Actividades de PARABLE

.

- Crear código QR, donde podemos crear nuevos QR para añadirlos en futuras historias.
- About, que da información sobre los miembros del proyecto y Valóranos, que envía *feedback* de los usuarios para poder mejorar la aplicación en un futuro.

Las opciones de Guardar Coordenadas y Crear Códigos QR no deberían ser usados por los usuarios finales pero los hemos dejado visibles para poder explicar su funcionamiento.

Parable esta desarrollado siguiendo el paradigma de Cliente-Servidor. Esta arquitectura nos permitía centralizar recursos, seguridad ante la integridad de los datos y facilitar el mantenimiento.

3.2. Cliente-Servidor

Para realizar PARABLE tomamos como base la arquitectura cliente-servidor. En nuestro caso, el servidor sería una máquina de la Facultad de Informática, llamada Tot, y prestada para llevar a cabo el proyecto. En este servidor se encontraría nuestra base de datos, nuestros archivos PHP que

funcionarían como gestores entre la aplicación móvil y la base de datos y algunos recursos visuales como videos o imágenes.

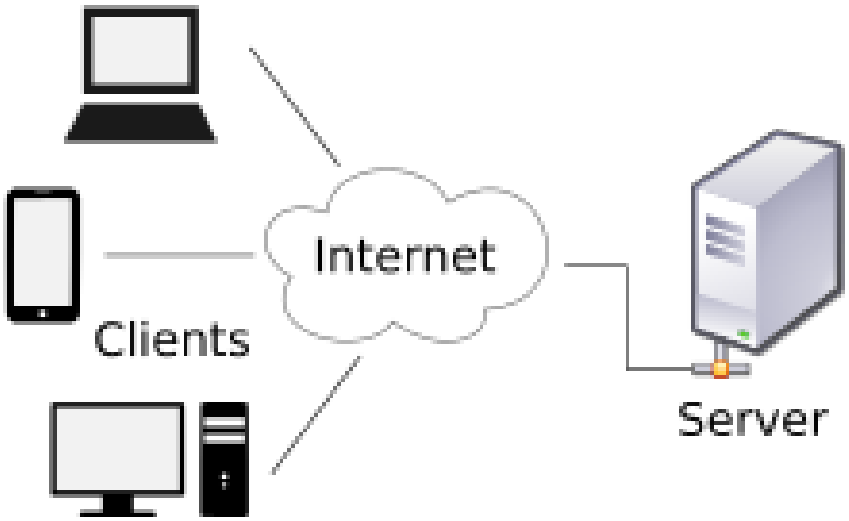


Figura 3.2: Arquitectura Cliente-Servidor.

Los clientes, es decir, los móviles que usen PARABLE, se conectan a nuestro servidor Tot a través de internet. Desde la aplicación se hacen peticiones al servidor. Estas peticiones pueden ser acceder a la base de datos para recuperar los datos necesarios para calcular su posición, introducir nuevas coordenadas, enviar la narración de su actividad o dar una valoración de nuestra aplicación. Todas estas peticiones se realizan usando PHP y sentencias *mysql*.

3.3. Posicionamiento

El posicionamiento *indoor* era uno de los pilares de nuestro proyecto ya que sin el no se podría narrar el camino del usuario. Para realizarlo se buscaron y se testearon varias opciones para ver cual era la mejor. En primer lugar, pensamos en usar el GPS propio del móvil. Descartamos rápidamente esta opción después de hacer una aplicación de prueba ya que el GPS no funciona bien, ni tiene la precisión suficiente para situarnos correctamente dentro del edificio. Otra opción, era que se usara el GPS mientras la señal fuera potente y en el momento que no lo fuera la aplicación cambiara a un posicionamiento por WiFi. Esta opción tampoco fue la elegida ya que nuestra aplicación se centraba en localizaciones interiores y era algo que en principio no usaríamos fuera de un edificio. Por último, se nos proporciono AVANTI (Enrique López Mañas, 2010) para que nos basáramos en su algo-

ritmo ya que en este proyecto se habían enfrentado a la misma dificultad. Esta fue la opción elegida finalmente y en el apartado 3.3.1 se explica más en profundidad.

3.3.1. Triangulación WiFi

Para realizar la triangulación mediante redes WiFi nos basamos en un trabajo (Enrique López Mañas, 2010) realizado anteriormente en esta misma facultad. Para adaptarlo a nuestras necesidades le realizamos algunos cambios usando funciones, tanto Java como Android, de versiones posteriores puesto que AVANTI se realizó en una de las primeras versiones de Android. La idea principal es recoger todas las intensidades de las WiFi's alrededor de nuestra posición y compararlas con las que se han mapeado anteriormente en la base de datos. Para ello usamos un sencillo algoritmo de los *K-Closest Neighbors*. Para el funcionamiento del algoritmo, primero debimos mapear la facultad. Se puede leer la explicación del mapeo en la sección 3.7.

3.3.2. K-Closest Neighbors

Este algoritmo calcula la posición del usuario a partir del cálculo de la distancia euclídea en la que usaremos como valores las intensidades y las coordenadas que han sido guardadas en la base de datos con anterioridad. El criterio usado para seleccionar las mejores posiciones es la distancia euclídea tomando como medidas las diferencias entre las intensidades de las señales. Si Z = (RSS_1,...,RSS_M) son el número de señales encontradas desde el terminal, compuesto por M puntos de acceso en una posición desconocida X = (x, y) (aquellas del terminal del cual deseamos obtenerla) y Z_i son señales registradas en la base de datos para la posición X_i = (x_i, y_i), entonces la distancia euclídea es:

$$d(Z, Z_i) = \frac{1}{M} \cdot \sqrt{\sum_{j-1}^{M} (RSS_j(x, y) - RSS_j(x_i, y_i))^2}$$

1. RSS_j(x_i, y_i): La fuerza de la señal recibida por el punto de acceso con la MAC j, localizado en x_i y y_i

2. M: número de diferentes puntos de acceso que se encuentran guardados en la base de datos.

El conjunto N_k de posiciones de la base de datos, que contiene el menor rango de error m en términos de RSS, se construye según el siguiente proceso iterativo:

$$N_k = \{argmin_{X_{i \in L}} [d(Z, Z_i)] \setminus X_i \notin N_k\}$$

Esta última fórmula recupera las K posiciones de la base de datos con el menor error, y sin ningún tipo de repetición. L es el conjunto de posiciones registradas dentro de la base de datos. Finalmente, la posición del dispositivo se calculará como el baricentro de las K posiciones:

$$X = \frac{\sum_{j=1}^{k}(1/d(Z, Z_i)) \cdot X_j}{\sum_{j=1}^{k}(1/d(Z, Z_i))}$$

Una de las principales ventajas de este algoritmo es la facilidad de implementación y uso. No obstante, la precisión depende de la cantidad de coordenadas de la base de datos de referencia, es decir, cuantas más coordenadas de las zonas de la facultad, más precisos serán los resultados obtenidos. Esto es debido a que a más intensidades en un mismo punto el cálculo será más preciso por tener más referencias y a su vez, a más coordenadas en la misma cantidad de espacio la posición será más exacta dentro de una misma situación o sección. Sin embargo, habría que tener cuidado con el número de accesos a la base de datos ya que esto puede ralentizar el uso de la misma. Para nuestro proyecto hemos tomado medidas cada 50 centímetros. Esta medida esta tomada pensando en la cantidad de coordenadas que se recogerán en cada paso de la persona que use la aplicación. Tuvimos en cuenta la posibilidad de que el usuario entrara en zonas de sombra (no se recibe señal WiFi) o zonas que no hubiéramos mapeado (como despachos privados o baños). Para solucionar esta situación, si el algoritmo no devuelve una coordenada válida, nosotros le asignamos el valor -1. Con este valor negativo indicamos a PARABLE que hay algún problema con la localización del terminal y aprovechamos esta situación para narrarla y decirle al usuario que ha entrado en una zona de sombra o no mapeada.

3.4. Generación de lenguaje natural

Nuestro objetivo en esta parte del desarrollo era conseguir una narrativa fluida y no repetitiva para conseguir una experiencia más interesante para el usuario. Para ello, creamos un fichero donde determinamos la acción realizada por un usuario o su situación y una narrativa añadida a esa acción o situación. De la misma manera, dentro de cada frase, y separado por corchetes, situamos aquellas partes que variarían en el caso de repetir la acción. Con esto conseguimos que al usar una plantilla y sustituyendo esas partes la narrativa no parezca la misma y se de la sensación de estar leyendo diferentes texto. (Ver Figura 3.3).

```
Saliendo_De=Salgo de [Lugar]
```

Figura 3.3: Ejemplo del fichero properties

Elegimos un mínimo de tres sinónimos para cada una de las palabras entre corchetes. Las razones por las que escogimos tres sinónimos al menos son conseguir esa variabilidad dentro de la narrativa, y porque no es probable que un usuario realice la misma acción más de tres veces seguidas. De todas formas en aquellos puntos seleccionado como puntos de interés (códigos QR y determinados por nosotros), tienen una narrativa más elaborada y con más opciones. En el caso en el que el usuario se quede quieto, la narración identifica que se está recibiendo la misma posición y se queda a la espera y la aplicación no repetirá la misma narración de la posición actual. Asimismo, si el usuario permanece más de tres veces en el mismo lugar, con PARABLE en marcha, se le indicará mediante una narración que está detenido. De la misma manera, si el usuario entra en zonas de sombra, como baños, despachos o zonas prohibidas al acceso, se le indicará automáticamente su situación y que debe volver a zonas mapeadas. Todas estas narraciones/locuciones tienen palabras que varían para no dar sensación de repetición. La evaluación de la repetición de acción, posición o intrusión en zona de sombra se realiza a la hora de recuperar la situación. Mención especial requieren las zonas no mapeadas o zonas de sombra. Al no estar incluidas las coordenadas en la base de datos, el algoritmo no puede triangular la posición. Esto lo hemos controlado de manera que al no encontrar una posición definida, el algoritmo devuelva -1. A continuación se evalúa esta coordenada y al ser negativa, PARABLE, identifica que no estamos en una zona disponible con la consecuente narrativa.

Como vemos en la Figura 3.3, algunas de estas palabras se encuentran entre corchetes se usarían para determinar el lugar en el que se encuentra el usuario. Para lograr colocar la posición adecuada, PARABLE, recupera la coordenada actual y la usa para buscar dentro de un archivo *xml* en el que tenemos la estructura de la facultad. Este archivo, llamado mapa.xml, se encuentra en el servidor y accedemos a él desde la aplicación. Como vemos en la Figura 3.4, el denominado como *pasillo1* se ha subdividido en tres secciones y como atributos, las coordenadas en las que se encuentra. Además, si la sección es un punto de interés se marca añadiendo una nueva etiqueta denominada *Poi*. Entonces, PARABLE recupera el nombre de este archivo a través de las coordenadas y lo sustituye entre los corchetes y para terminar, se le muestra al usuario. Hay varias palabras que van entre corchetes en cada acción, y son estas las que se variarán, como se ha dicho antes, para dar variabilidad a la narrativa.

```
<Pasillo inix="0" finx="3" iniy="0" finy="67">
    <NombrePasillo> Pasillo1 </NombrePasillo>
        <Seccion inix="0" finx="3" iniy="0" finy="22">
            <NombreSeccion> InicioPasillo1 </NombreSeccion>
        </Seccion>
        <Seccion inix="0" finx="3" iniy="23" finy="44">
            <NombreSeccion> MedioPasillo1 </NombreSeccion>
        </Seccion>
        <Seccion inix="0" finx="3" iniy="45" finy="67">
            <NombreSeccion> FinalPasillo1 </NombreSeccion>
            <Poi>1</Poi>
        </Seccion>
</Pasillo>
```

Figura 3.4: Estructura de un pasillo

Si durante el uso de la aplicación nos encontramos con un código QR podremos escanearlo simplemente haciendo *clic* en la opción adecuada. Para esta opción hemos usado la librería *ZXing*[4] que nos permite gestionar la lectura y creación de estos códigos. Al leer el código QR recuperamos de su información, tanto el valor del código, como la narrativa del mismo que se añadirá a nuestra historia. Para realizar esto, *ZXing* nos ofrece la posibilidad de recuperar los datos y a continuación con un simple parseo recuperar la información que necesitamos.

Por último, todas las narrativas se van mostrando secuencialmente en la actividad de la aplicación. Se muestra en un *Scroll* porque las historias podrían ser lo suficientemente largas como para no caber en la pantalla del ordenador. Hemos automatizado el *Scroll* para que siempre muestre las últimas frases añadidas y no se quede fijo en las primeras lineas. Mientras usamos la aplicación, las frases que cuenta nuestra historia son independientes unas de otras, así que, para solucionarlo, las vamos añadiendo en una variable *String* que será lo que se muestre en la actividad.

[4]https://github.com/zxing/zxing

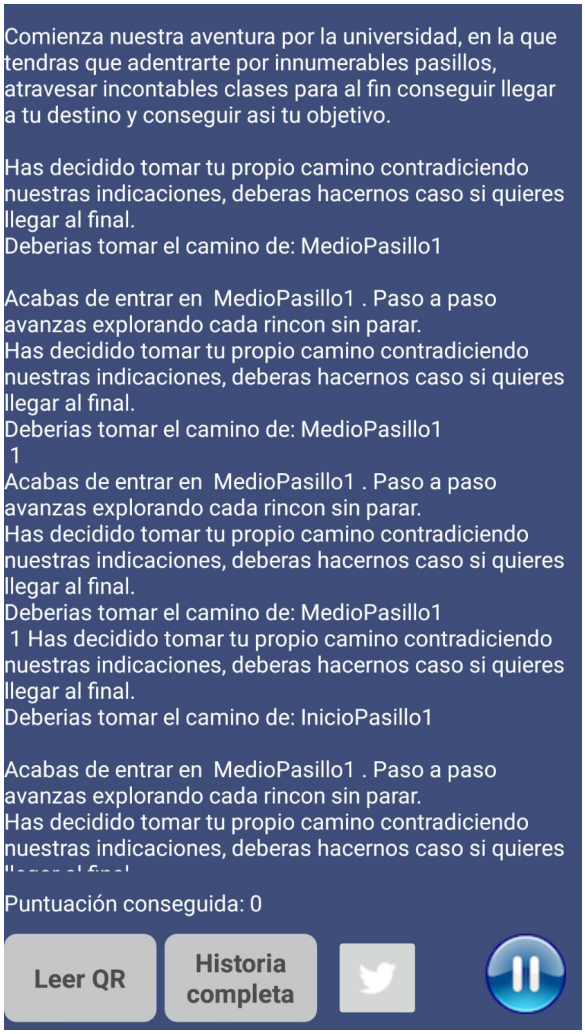


Figura 3.5: Generación de Lenguaje Natural

Al realizar PARABLE, hemos tenido en cuenta que podría ser tedioso estar leyendo toda la información que sale por la pantalla del móvil. Para solucionar este problema dotamos a la aplicación de tecnología *text to speech*. Esta tecnología es un tipo de síntesis de voz que se utiliza para crear una versión audible de un texto. Normalmente son textos de ayuda o informativos. El *text to speech* puede permitir la lectura de información de la pantalla de nuestro dispositivo para personas con discapacidad visual, o puede simplemente ser utilizado para la lectura de un mensaje de texto. Algunas de las aplicaciones actuales que incluyen *text to speech* son los gestores de correo electrónico o sistemas operativos móviles para proporcionar accesibilidad. En nuestro caso, esta tecnología está proporcionada nativamente por Android y nos ofrece la posibilidad de narrar mediante audio las frases y textos

que va generando la aplicación. Para llevarlo a cabo usamos la librería *android.speech.tts.TextToSpeech* y concretamente el método *speak()* al que le pasamos el texto mediante una cadena. Anteriormente, hemos configurado el lenguaje de la aplicación a castellano. También se pensó que ofrecer el audio de todo el texto generaría confusión y molestia, así que, solo se aplicaría esta tecnología a las secciones de la facultad denominados como puntos de interés, y además, dentro de toda la narrativa generada en estos puntos, solo se daría audio a la primera frase que es la que muestra la posición.

En futuras actualizaciones se pretende añadir más opciones de lenguajes, volumen o elegir diferentes tipos de voz pero debido al escaso tiempo disponible no ha sido posible ejecutar estas funcionalidades.

3.4.1. Lectura de códigos QR

Cuando el usuario se encuentra con un código QR, este tendrá la opción de leerlo. El código contiene información de la posición en la que se encuentra que pasará a formar parte de la narrativa. Al estar en un punto fijo, estos códigos nos han permitido dotar a la historia de un texto más enriquecido. Además, añade diez puntos a la puntuación del usuario.

La forma de realizar esta funcionalidad es la siguiente: los códigos han sido previamente cargados en la base de datos y cuando el usuario encuentra uno, activa la opción de leer QR, la aplicación conecta con la base de datos y comprueba, mediante el campo *Leido*, si el usuario ya ha registrado el código. Si es nuevo, añadirá su información a la historia y en el caso en el que se lea más de una vez, o que no exista, se mostrará un aviso de la situación. La conexión con la base de datos se hace a través de un nuevo hilo (*Asyntask*) proporcionado por la aplicación que conecta mediante *httppost* con nuestra base de datos. Allí, se hace mediante el archivo PHP *getQr.php* una busqueda para comprobar si existe el código y además si ha sido o no leído. Después se devuelve la información y se muestra por la pantalla de la aplicación. Por último, con el fin de proteger la creación y lectura de los QR, su información ha sido encriptada en la base de datos mediante la función PHP *md5()* que utiliza el algoritmo MD5[5] de 128 bits. Se puede ver la tabla que usamos para esta funcionalidad en la Figura 3.12.

3.4.2. Redes sociales

Las redes sociales son un gran modo de entretenimiento y una ventana al mundo. Queríamos que esto se viera reflejado en nuestra aplicación. Actualmente, Twitter[6] es una de las mayores fuentes de información en tiempo real en internet. En 140 caracteres se pueden enviar mensajes claros y directos con suficiente información como para que puedan ser tenidos en cuenta.

[5] http://php.net/manual/es/function.md5.php
[6] http://www.twitter.com

Twitter proporciona una API abierta para todo tipo de desarrolladores, lo cual supone una gran ventaja para todos aquellos que quieran integrarlo como servicio en otra aplicación. Hemos usado esta API para llevar a cabo la funcionalidad de enviar la narrativa más allá de la propia aplicación. Para ello, realizamos un *tweetbot* en el que al encontrarnos en una situación determinada como punto de interés tuiteara la acción que estaba haciendo el usuario. El funcionamiento es bastante simple. Cuando el usuario accede a un punto de interés o *checkpoint* se avisa a la función del *bot* que tuiteara la situación actual apoyándose en la clase *SocialNetwork*, concretamente en el método *twitter()*. Por otro lado queríamos ofrecer al usuario la posibilidad de tuitear en cualquier momento, así que añadimos un botón a la interfaz principal que permitiera realizar dicha acción.

Otra funcionalidad que añadimos fue el poner a disposición de otros usuarios (o cualquier persona) la historia que se había vivido. Para ello una vez que la narrativa ha terminado, se ofrece la opción de enviar por mail la historia o si se prefiere enviarse a la pagina web de PARABLE: `http://tot.fdi.ucm.es/parable/parableweb/index.php` que está almacenada en nuestro servidor. Para conseguir esta funcionalidad, una vez que se ha terminado la narrativa, si el usuario hace uso del envío a la web se activa una función de la clase *HttpService* y del mismo modo que al insertar una coordenada, insertamos esta historia en la base de datos. Si el usuario prefiere enviarlo por mail, se llama a la clase *SocialNetwork* y dentro de ella al método *correo()* que al pasarle una dirección de correo se enviaría desde tu gestor de correo personal. El resultado se puede ver en la Figura 3.6.

Por último, quisimos dar la posibilidad a los usuarios de ofrecernos un *feedback* de como ha sido su experiencia con la aplicación. Por esta razón incluimos una actividad dentro de la aplicación que nos ofreciera esta posibilidad. El funcionamiento es análogo al proceso de enviar la historia a nuestra base de datos. El usuario escribe su nombre y una breve reseña y puntúa la aplicación de uno a cinco que luego será almacenado a través de un archivo php (insertar .php) en la base de datos (Figura 3.11).

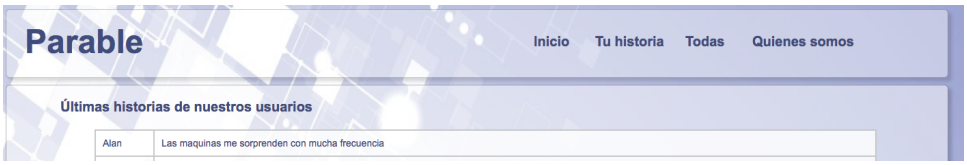


Figura 3.6: Página principal de la web de PARABLE

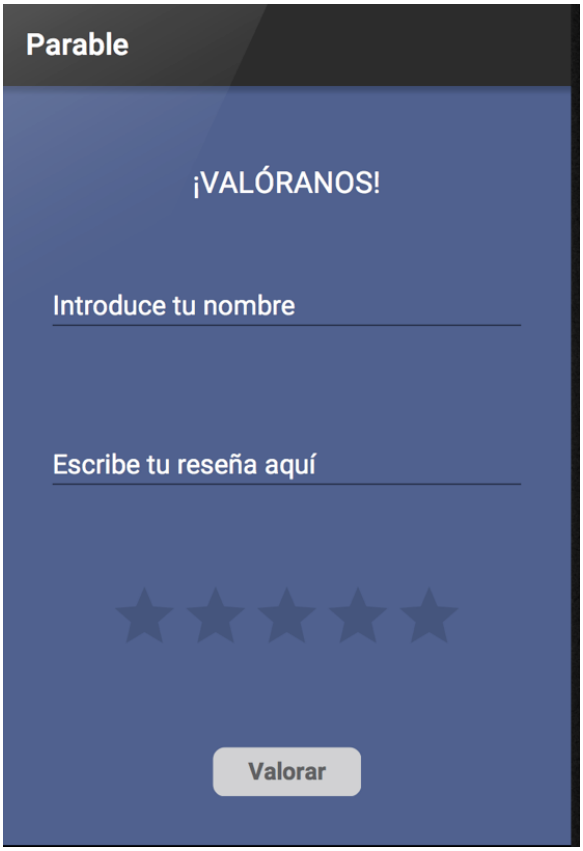


Figura 3.7: Actividad que permite valorar PARABLE

3.5. Base de Datos

La base de datos de nuestra aplicación consta de cuatro tablas simples que no tienen relación entre sí. El acceso a la misma se hace mediante archivos PHP y sentencias mysql. Está instalada en el servidor Tot que nos ha prestado la facultad que siempre está encendido por lo que el acceso siempre debería estar disponible. Dentro de *mysql* optamos por sentencias *mysqli* ya que nos proporcionaba más seguridad frente a SQL injections que las *mysql* y además nos permitía una interfaz orientada a objetos, soporte para transacciones y *rollbacks* y mejores opciones de depuración. Elegimos la arquitectura cliente-servidor basándonos en las opciones que ofrecía frente a tener toda la base de datos dentro de la aplicación, lo cual hubiera aumentado el tamaño de la misma y la lentitud a la hora de procesar la información. Las ventajas de cliente-servidor eran conseguir una centralización de los recursos usados por la aplicación que nos llevaría a poder mantener la integridad de los mismos. Es decir, si hay algún dato corrupto o un error en un PHP solo tendríamos que modificarlo en el servidor y no lanzar una actualización para arreglarlo

en todas las aplicaciones. Por lo tanto, el mantenimiento también se reducía solo a la parte del servidor. Una de las desventajas que valoramos a la hora de elegir esta arquitectura fue la preocupación de que si el servidor no estaba operativo, la propia aplicación dejaría de funcionar. Seguimos adelante teniendo en cuenta que es un proyecto de fin de grado y que esta casuística era improbable. Sin embargo habría que tenerlo en cuenta si se tratará de un proyecto comercial. Otro de los problemas que tuvimos en cuenta fueron la cantidad de conexiones que se pudieran hacer a la base de datos y si se pudiera dar el caso de que ocurrieran *deadlocks* o actualizaciones de la base de datos no permitidas. Para solucionarlo, convertimos todos los campos de la base de datos en claves primarias salvo el nivel de intensidad. Así, si se daba una actualización de una coordenada ya incorporada a la base de datos, esta no sería posible. En cuanto al número de accesos a la base de datos, no era preocupante ya que aunque se realizarán pruebas con varios dispositivos a la vez, Tot podía soportar el nivel de masa crítica de clientes en la escala de la facultad. La actividad desde la cual podemos enviar las coordenadas para que sean introducidas en la base de datos la podemos ver en la Figura 3.26.

A continuación se da el esquema de las tres tablas y una descripción de las mismas:

- La tabla *coordenadas* almacena las coordenadas que han sido mapeadas previamente por nosotros. Por seguridad hemos definido como claves primarias las coordenadas X,Y, y Z además de la MAC para que si se registra una coordenada que ya estaba definida en la base de datos, esta no lo permita. Así conseguimos que no haya duplicidad de coordenadas y una mayor seguridad.

- La tabla *historia* almacena las historias del usuario si este lo desea (hay una opción de almacenar la historia en la base de datos). Esta historia podrá luego ser recuperada mediante la visita a la página web de nuestra aplicación.

- La tabla *valoracion* contiene las valoraciones hechas por los usuarios de la aplicación y los comentarios que nos puedan servir para mejorar.

- La tabla *Qr* contiene las la información de los códigos QR encriptados y si han sido leídos por el usuario o no.

3.6. Servicios web

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. En PARABLE, usamos estos servicios web para comunicarnos con el servidor, la base de datos y con

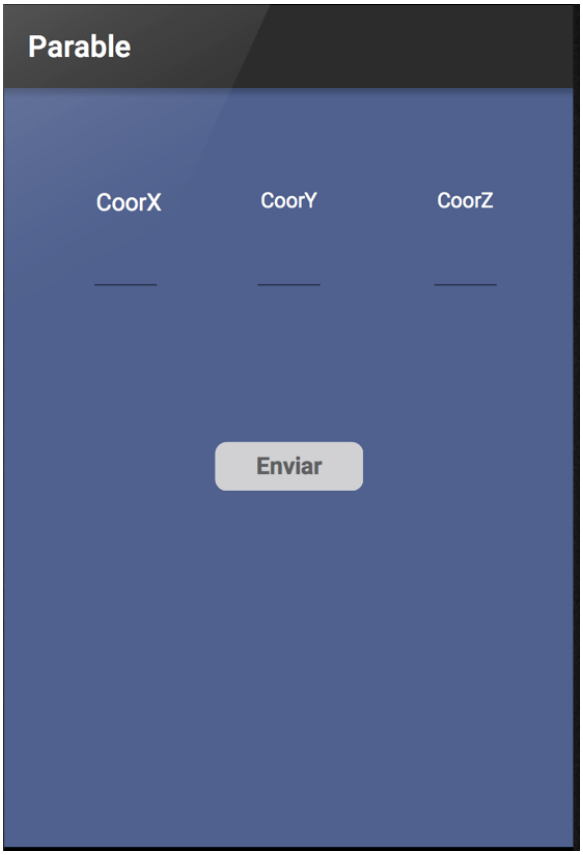


Figura 3.8: Actividad que permite introducir las coordenadas en la base de datos

Coordenadas	
MAC	varchar(17)
X	double
Y	double
Z	double
Nivel	int(11)

Figura 3.9: Tabla Coordenadas.

los archivos *XML*. Concretamente, los archivos necesarios para conseguir las posiciones (*getAllPositions.php*), insertar las coordenadas en la base de datos

Historia	
id	int(8)
nombre	varchar(20)
texto	varchar(5000)
valoración	double

Figura 3.10: Tabla Historia.

Valoración	
id	int(8)
nombre	varchar(20)
resena	varchar(1000)
valoracion	double

Figura 3.11: Tabla Valoración.

Qr	
id	int(8)
fraseEnc	varchar(200)
Leido	int(10)

Figura 3.12: Tabla Qr.

(*insertarEnBD.php*) y recuperar la posición en lenguaje natural (*mapa.xml*). Estos archivos son llamados desde la aplicación a través de sentencias *http-Post*. Podemos encontrar las funciones que comunican la aplicación con el servidor en el paquete http y esas serán usadas a través de hilos secundarios ya que en el hilo principal produce un error. Este error viene provocado porque Android no permite que el hilo principal quede bloqueado. Por último señalar que estos servicios se comunican a través del puerto 80.

3.7. Mapeo

Para llevar a cabo las pruebas de posicionamiento, en primer lugar mapeamos un solo pasillo de la primera planta de la facultad de FDI. Elegimos esta planta porque las intensidades de las redes WiFi eran las más potentes (puntos verdes de la Figura 3.14)y así tendríamos resultados más estables. Dividimos el pasillo en tres zonas (inicio, medio y final) para situar a la persona y mapeamos cada una de ellas. Una vez que las pruebas de posicionamiento nos dieron resultados correctos, mapeamos el resto de la primera planta incluyendo las aulas. Al realizar el algoritmo de los K-Closest Neighbors nos dimos cuenta de que cuantas más coordenadas incluyéramos en la base de datos más real y preciso sería el resultado del posicionamiento. En contraposición, si una coordenada tenía muchas redes WiFi asociadas el funcionamiento de la aplicación se vería ralentizado por los accesos a la base de datos del lado del servidor. Así que la solución propuesta fue mapear cada 50 centímetros (algo menos de una zancada humana). Con estas medidas conseguimos unas 4 coordenadas de ancho en cada pasillo, que serían nuestras X, y 67 coordenadas de largo en cada pasillo, que serían nuestras Y. La Z en este caso sería siempre 1. Además, hay que tener en cuenta que en cada coordenada recuperábamos varias señales WiFi que suponían, cada una de ellas, una entrada en la base de datos. El proceso de mapeado fue largo ya que solo disponíamos de un móvil para realizarlo y teníamos que recoger todas las intensidades, darles coordenadas e introducir en la base de datos para cada una de nuestras coordenadas. El proceso duró varios días para el primer piso. Al final del proceso de mapeo, contando con los dos pasillos largos, los dos cortos y las aulas dieron como resultado la introducción de entre 14000-15000 coordenadas.

Si tenemos en cuenta la parte técnica, guardamos las coordenadas en la base de datos mediante archivos PHP. Al pulsar el botón de guardar coordenada, la aplicación, que se apoya en la arquitectura cliente-servidor, hace una llamada al archivo PHP *insertarEnBD.php* que se encarga mediante sentencias *mysql* de hacer la inserción en la base de datos de las propias coordenadas, la MAC y la intensidad.

3.8. Cálculo del camino al destino fijado por la aplicación

PARABLE interpreta la facultad de FDI como si se tratara de un grafo no dirigido. Es decir, cada sección de la misma es un vértice y las aristas son las conexiones entre las diferentes aulas, pasillos o plantas. Para realizar esta característica, hemos utilizado la librería jGrapht[7] que nos permite

[7] http://jgrapht.org/

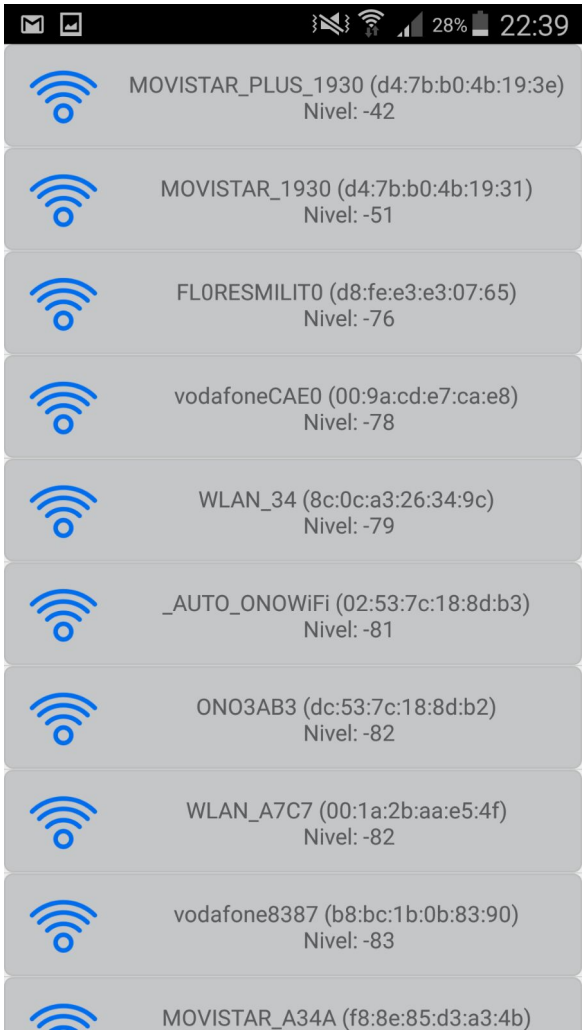


Figura 3.13: WiFi's que encontramos alrededor del dispositivo.

construir, manejar y gestionar el grafo de manera simple. En la construcción del grafo hemos definido los vértices como *strings*, lo cual nos permite comparar la posición del usuario frente al grafo y así mostrar las opciones de avance en la historia. Para lograr mostrar los diferentes caminos que puede tomar en cada momento, la librería jGrapht, nos ofrece la función de mostrar los vecinos de un vértice (*neighborListOf(vértice)*). Así, teniendo esta información, comparamos si el usuario esta siguiendo el curso natural de la narración o por el contrario desobedece. Además, cada vez que llegamos a un nuevo nodo, recuperamos el camino más corto para llegar a nuestros objetivos, que pueden ser varios. Esto lo conseguimos aplicando el algoritmo de *Dijkstra* que se puede ver en la Figura 3.15. Este algoritmo siempre nos muestra el camino óptimo para llegar al siguiente *checkpoint* y en cada vér-

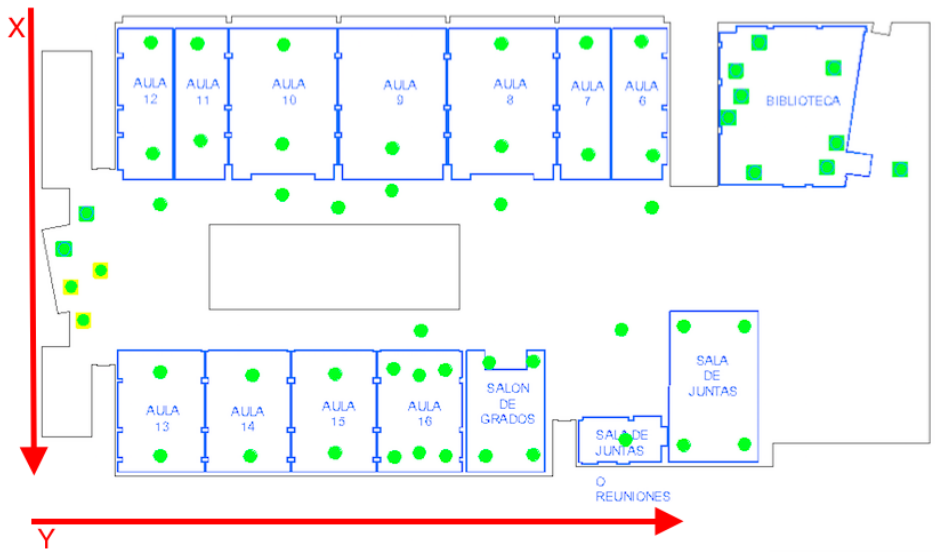


Figura 3.14: Mapa de la primera planta de FDI

tice nuevo al que lleguemos nos mostrará cual es nuestra mejor opción para continuar. Este algoritmo lo hemos recuperado de la misma librería usando la función *DijkstraShortestPath(grafo,vértice inicio, vértice final)*. En la aplicación, cada vez que alcancemos un *checkpoint* cambiaremos el vértice final por el nuevo *checkpoint*. Así hasta terminar con la lista de *checkpoint* la cual tendrá como ultimo elemento el punto final de la historia. Como vértice inicial incluimos aquel en el que nos encontramos en el instante actual.

Esta es la forma que tenemos de controlar si el usuario obedece a la narración y en función del camino elegido mostrar unas frases u otras. Si sigue el camino marcado por *Dijkstra*, la historia seguirá su curso normal. Si por el contrario el vértice elegido por el usuario no es el elegido por *Dijkstra*, mostrará una frase que denotará que no se ha seguido la narración de la aplicación. Por último y para no hacer muy repetitiva la historia, se ha utilizado un sistema de sinónimos que cambia cada vez que una de las frases de la historia se repite.

```
función Dijkstra (Grafo G, nodo_salida s)
  //Usaremos un vector para guardar las distancias del nodo salida al resto
  entero distancia[n]
  //Inicializamos el vector con distancias iniciales
  booleano visto[n]
  //vector de boleanos para controlar los vértices de los que ya tenemos la distancia mínima
  para cada w ∈ V[G] hacer
     Si (no existe arista entre s y w) entonces
         distancia[w] = Infinito //puedes marcar la casilla con un -1 por ejemplo
     Si_no
         distancia[w] = peso (s, w)
     fin si
  fin para
  distancia[s] = 0
  visto[s] = cierto
  //n es el número de vértices que tiene el Grafo
  mientras que (no_estén_vistos_todos) hacer
     vértice = coger_el_mínimo_del_vector distancia y que no esté visto;
     visto[vértice] = cierto;
     para cada w ∈ sucesores (G, vértice) hacer
         si distancia[w]>distancia[vértice]+peso (vértice, w) entonces
            distancia[w] = distancia[vértice]+peso (vértice, w)
         fin si
     fin para
  fin mientras
fin función.
```

Figura 3.15: Pseudocódigo del algoritmo Dijkstra

3.9. Diagramas y Patrones

3.9.1. Patrones

Desde el principio quisimos tener una claridad en el código fuente para conseguir un acomplamiento bajo y una cohesión alta de los paquetes y características. Para ello, desde el inicio programamos siguiente los siguientes patrones de diseño:

- **Singleton:** Es un patrón diseñado para restringir la creación de objetos pertenecientes a una clase. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. En nuestra aplicación lo usamos para tener solo una instancia del controlador, la factoría y el dispatcher. El patrón singleton lo tenemos definido en la Figura 3.16.

- **Factoría abstracta:** Es un patrón diseñado para cuando se desea incluir nuevas familias de componentes que dependen mutuamente y sin especificar cual es el objeto concreto. En nuestra aplicación lo usamos para crear las familias de comandos. El patrón factoría lo tenemos definido en la Figura 3.17.

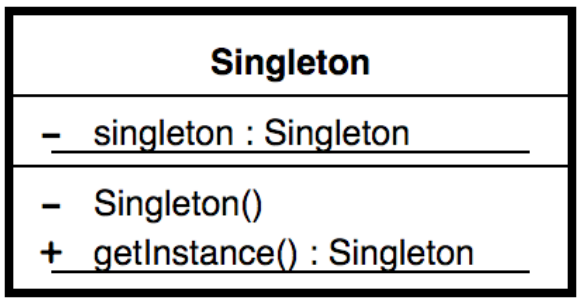


Figura 3.16: Patrón Singleton

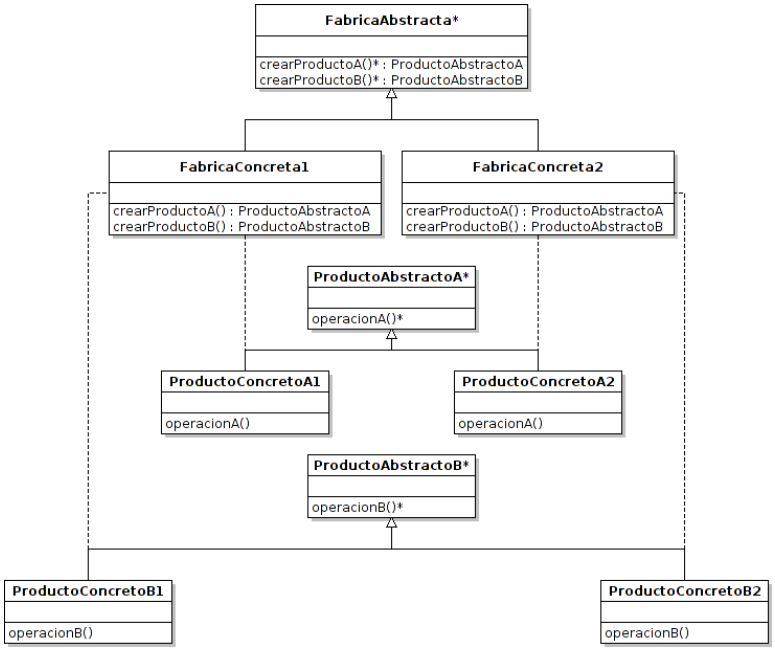


Figura 3.17: Patrón Factoría Abstracta

- **Comando:** Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de la operación, ni el receptor de la misma. Para ello se encapsula la petición como un objeto, con lo que además facilita la parametrización de los métodos. En nuestra aplicación lo usamos para marcar las acciones del usuario, es decir, si esta andando, parado, entrando en determinado sección, etc. El patrón comando lo tenemos definido en la Figura 3.18.

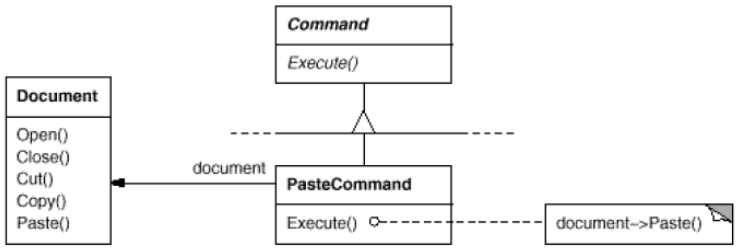


Figura 3.18: Patrón Comando

- **MVC:** El patrón modelo-vista-controlador separa los datos y la lógica de la aplicación de la interfaz. El modelo es la representación de la información con la cual el sistema opera. El controlador responde a los eventos, que en nuestro caso son los comandos o acciones del usuarios. Por último, la vista presenta un modelo en un formato adecuado para actuar. En PARABLE, hemos querido diferencias claramente aquellas clases que son actividades (vista), de aquellas que se ocupan de la gestión de la información (modelo) y también de los comandos (controlador) ejecutados por el usuario. El patrón MVC lo tenemos definido en la Figura 3.19.

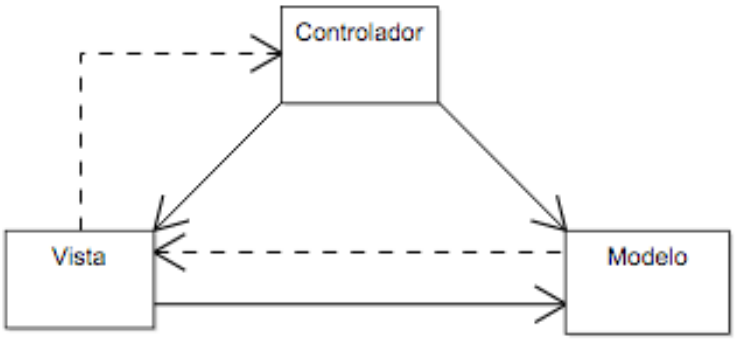


Figura 3.19: Patrón Modelo-Vista-Controlador

3.9.2. Descripción de componentes

En esta sección se muestran los diagramas de clases de las diferentes partes de PARABLE. Además se dará una visión general del funcionamiento de cada clase.

La siguiente figura muestra el patrón Comando descrito en la sección 3.9.1.

- La clase Comando es de tipo interfaz y contiene un solo método. Este método es el *ejecutar()* que será desarrollado por todos aquellos co-

mandos que implementen la interfaz. En cada uno de los comandos (AndandoEn, EntrandoEn, etc), se le pasa al método *ejecutar()* un *IDEventos* que identifica el evento que está realizando el usuario y unos datos que provienen de la clase *Lenguaje* y dependen también del evento en cuestión. Una vez ejecutado el comando devuelve un objeto de tipo *ComandoResponse*

- Las clases *ComandoFactoria*(interfaz) y *ComandoFactoriaImp* son aquellas que se encargan de la creación del comando dependiendo del identificador de comando que se le envíe.

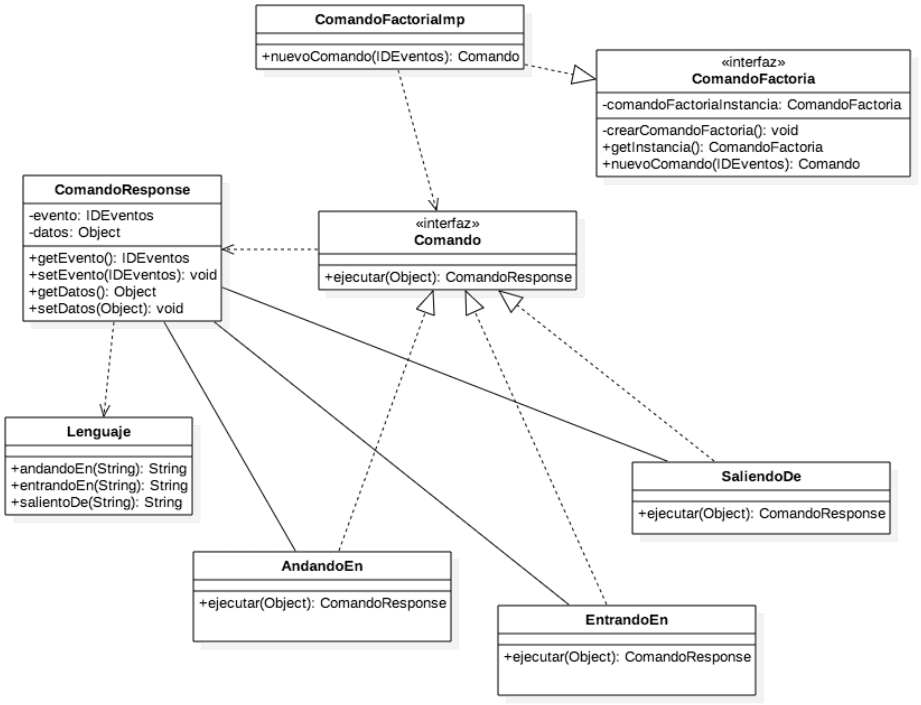


Figura 3.20: Diagrama de patrones Comando y Factoria

La siguiente figura muestra el patrón Controlador descrito en la sección 3.9.1.

- Las clases *Controlador*(interfaz) y *ControladorImp* son aquellas clases que hacen la petición de crear un nuevo comando, lo cual se ha visto en el anterior diagrama, es decir, el método *handleRequest()*. Este método también llama al *Dispatcher*

- La clase *Dispatcher* y su implementación reciben la respuesta después de ejecutar el comando y recuperan los datos y el evento que a continuación se le envían a la clase *ParableGUI* para su actualización.

- Las clases *ParableGUI* y *ParableGUIIMP* se ocupan de la actualización y de la interfaz (a la que se le envían los datos narrativos) y añade la narrativa a la variable donde se almacena la información narrativa.

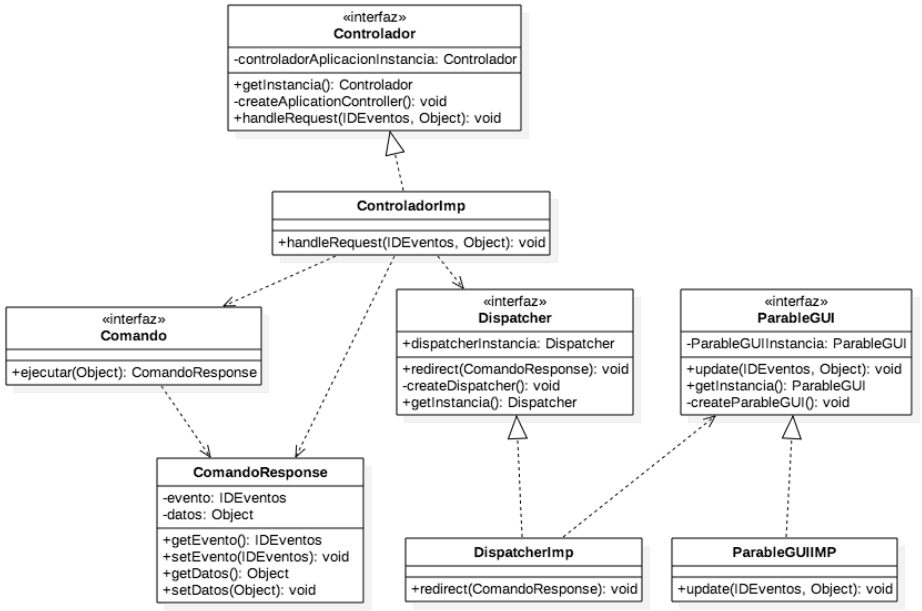


Figura 3.21: Diagrama del patron MVC

En la Figura 3.22 tenemos el diagrama principal de la aplicación. Toda la aplicación gira alrededor de las posiciones del usuario en cada momento.

- La clase *WPSDatabase* es una clase importante ya que se ocupa de insertar o recuperar las coordenadas de la base de datos. Es importante explicar los métodos de esta clase para poder comprender mejor su funcionamiento. En primer lugar tenemos *insertarCoordenadas()*. Este método construye una *query mysql* a partir de las coordenadas y la ejecuta sobre una base de datos *SQLite*. El siguiente método es *getCoordenadasEnPunto()* que se ocupa de recuperar todas las coordenadas que pueden ser encontradas en una determinada posición en la base de datos. Luego recorremos un cursor sobre todas las filas encontradas y las añadimos a un *array* de coordenadas. En este método hemos controlado que guardaríamos aquellas coordenadas que tuvieran una intensidad mayor de -83. Este valor lo hallamos a partir de la media de todas las intensidades y con el fin de poder quedarnos con las que nos ofrecían una mayor seguridad. El siguiente método es *getClosestNeighbors()* que recupera los vecinos más cercanos a partir de una lista de direcciones *MAC*, es decir, por cada una de las direccio-

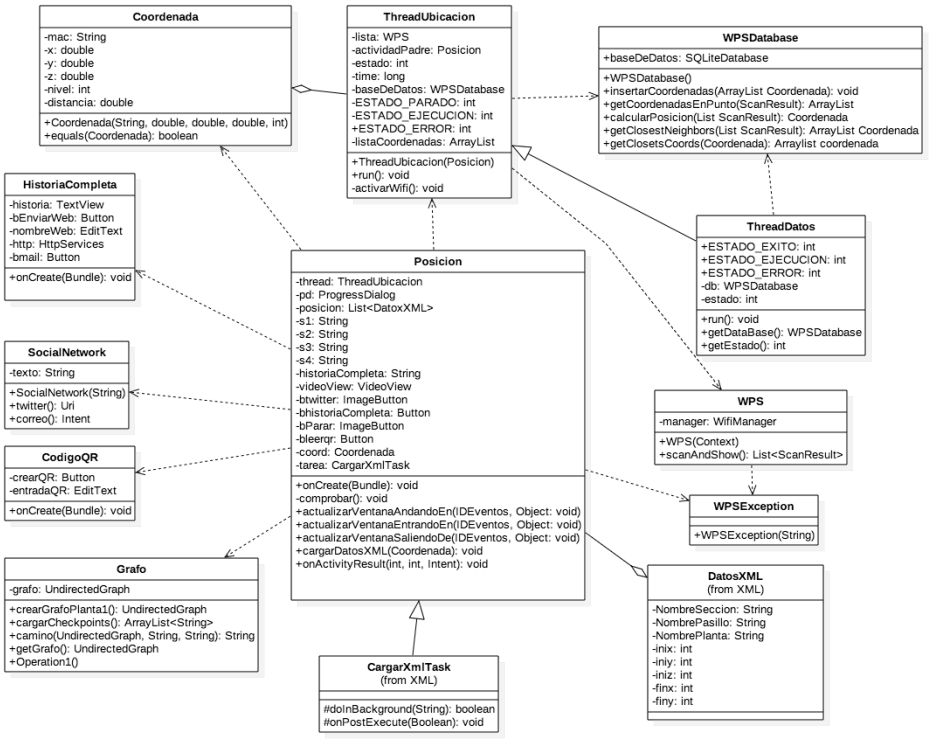


Figura 3.22: Diagrama principal

nes de la lista, recupera todos sus vecinos cercanos y se queda con los diez mejores. A continuación tenemos *getClosetsCoords()* que recupera las coordenadas más cercanas a una dada. El método para hallarlas es análogo a los anteriores. Se realiza una *query* y a partir de un cursor se recorren las filas. La distancia se calcula a partir de las distancias euclídeas entre coordenadas.

- La clase anterior se apoya en la clase *ThreadDatos* para poder crear un hilo que no interrumpa al hilo principal y así poder insertar los datos con mayor fluidez.

- Una clase de gran importancia en PARABLE es *ThreadUbicacion*. Cuando la aplicación necesita recuperar la posición del usuario, esta crea un hilo desde esta clase que, en el caso en el que el WiFi del móvil no este conectado, usa la clase *WPS* que activa esta función. Esta clase crea un hilo de ejecución y al llamar a su método *run()* se pone en funcionamiento el calculo y recuperación de la posición del usuario. Todo estos cálculos se hacen a través de las siguientes sentencias *coord = MainActivity.threadDatos.getDatabase().calcularPosicion(lista.scanAndShow()); coord = stimation(coord); coord = miRedondeo(*

coord);. La primera calcula la posición a través de un nuevo hilo creado por *ThreadDatos* que recupera la base de datos y calcula la posición después de pasarle el escaneo de las intensidades en ese momento y posición. La siguiente sentencia hace una estimación de las coordenadas y por último un redondeo de la propia coordenada para afinas el resultado y estandarizarlo. Este método esta ejecutándose continuamente mientras la aplicación este en funcionamiento. Toda la información es traída de la base de datos y las clases *DatosXML* y *CargarXmlTask*, y se formatea para mostrar la narrativa propia de la aplicación.

- La clase *Posicion* es la clase de tipo actividad que da interfaz a todo el cálculo de la posición del usuario y muestra la narrativa. Además ofrece posibilidades como ver la historia completa (clase *HistoriaCompleta*), enviar la historia al mail, enviar un *tweet* (apoyados en la clase *SocialNetwork*)

- Otra clase importante es la clase *Grafo* que crea un grafo no dirigido que emula a la facultad y calcula el camino más rápido para llegar al siguiente *checkpoint*. Además comprueba si el camino elegido es el correcto o no y añade información que luego será utilizada para completar la narrativa.

- Por último, tenemos la clase *CodigoQR* que se ocupa de la lectura y creación de estos códigos. Para esta clase nos hemos apoyado en una librería externa llamada *ZXing* `https://raw.github.com/embarkmobile/zxing-android-minimal/mvn-repo/maven-repository/`

En la Figura 3.23 tenemos la clase *WPSActivity* y la clase *WPS*

- La clase *WPSActivity* se ocupa de recuperar todas las redes WiFi disponibles que el terminal encuentra alrededor del usuario. Es una clase actividad con lo cual es parte de la interfaz de la aplicación. En este caso muestra una lista de las WiFi disponibles, es decir, su nombre, su dirección MAC y su intensidad. Para ello se apoya en la clase *WPS*

- La clase *WPS* se ocupa del escaneo de las redes a través de un *WifiManager*.

En la Figura 3.24 tenemos la representación de como enviar y recuperar datos desde la base de datos.

- La clase *WPSDatosBD* es la clase principal se ocupa de esta característica. Usa las clases *Coordenada* que es aquello que vamos a insertar a través de la clase interna *InsertarClase* y con ayuda de *HttpServices* que es la clase que enlaza con los *php* del servidor. Es una clase actividad, con lo cual forma parte de la interfaz de la aplicación. En esta actividad podemos darle valor a las coordenadas mediante unos campos *EditText* e insertarlas en la base de datos.

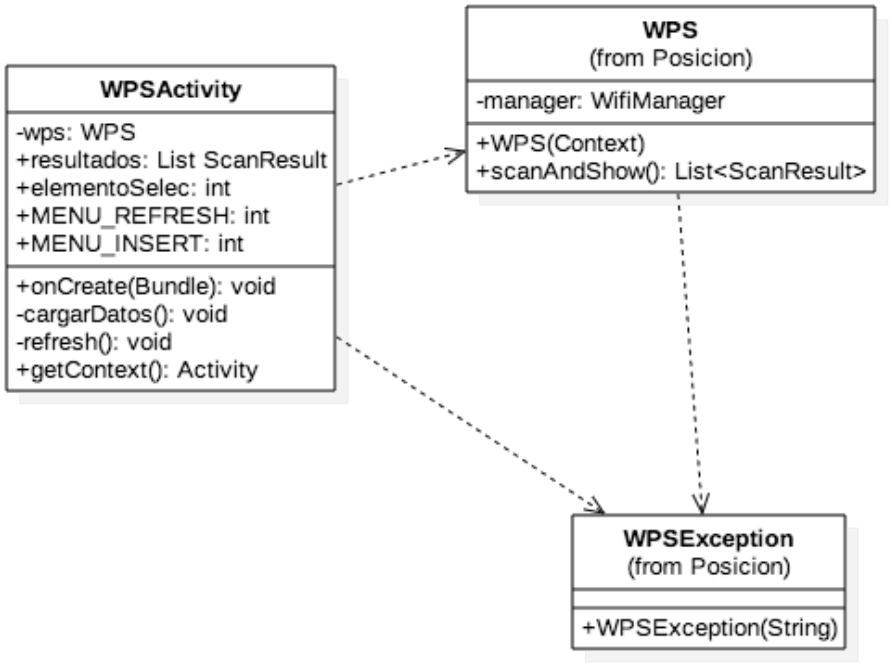


Figura 3.23: Diagrama de la conexión y comprobación del estado de la conexión WiFi

- La clase interna *InsertarClase* es la que se ocupa de realizar la acción de inserción a través de la siguiente sentencia: *services.insertarBD(resultados.get(i).BSSID, xEdit, yEdit, zEdit, resultados.get(i).level).*

- La clase *HttpServices* es clase que contiene los métodos que se ocupan de la inserción (*insertarBD()*) en la base de datos. A través de una lista de pares de valores, donde le damos un nombre al dato que vamos a insertar y a continuación el valor que queremos insertar (*BasicNameValuePair("mac", macEdit)*). Hay que tener en cuenta, también la sentencia *HttpPost* que es aquella que conecta los datos de la aplicación móvil con el archivo *PHP* que realizará el *insert*. Cabe señalar que esta función solo funcionará si se encuentra llamada por un hilo secundario ya que si lo hacemos en el hilo principal nos dará un error ya que puede hacer que se ralentice mucho la ejecución. Otro método de esta clase es *recuperarCOordenadasEnString()* que convierte las coordenadas en una cadena y es usado internamente por el método *getCoordenadas()* que como su nombre indica recupera las coordenadas. Por último, tenemos el método *enviarHistoria()* que análogamente al método *insertarBD()*, inserta la historia narrativa en la página web.

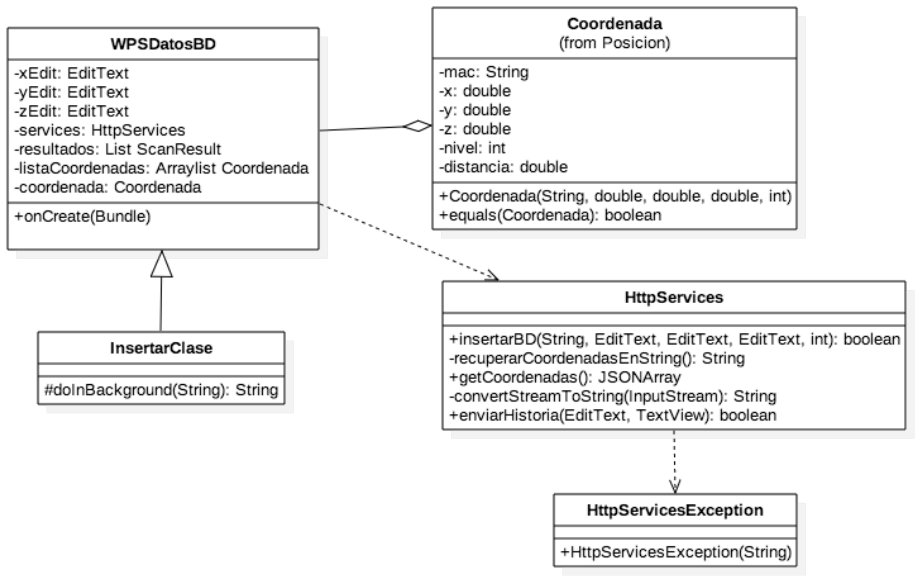


Figura 3.24: Diagrama del envío y recuperación de datos de la base de datos

En la Figura 3.25 tenemos las clases que se encargan de la recuperación de los datos de posición. Estos datos se encuentran en un archivo *xml*.

- *CargarXmlTask* es una clase interna de la clase *Posicion* que se, a partir del dato recuperado desde el fichero XML, evalúa si estamos siguiendo el camino correcto o estamos desobedeciendo. Esta clase interna se apoya en el cálculo de Dijkstra de la clase *Grafo*.

- La clase *DatosXML* es un simple transfer que almacena la información de la posición del usuario recuperada del archivo *XML* después de tener la coordenada.

- La clase *RssParserSax* es la clase que recibe la coordenada y la parsea para enviársela a la clase *RssHandler*

- Por último, la clase *RssHandler* recibe los datos de la coordenada parseados por la clase *RssParserSax* y busca dentro del fichero *XML*, comparando las coordenadas con los atributos del fichero, la posición del usuario en lenguaje natural.

3.10. Página web

Hoy en día es difícil ver una aplicación para móvil que no tenga una página web de presentación de la aplicación y, que además, añada algún tipo de

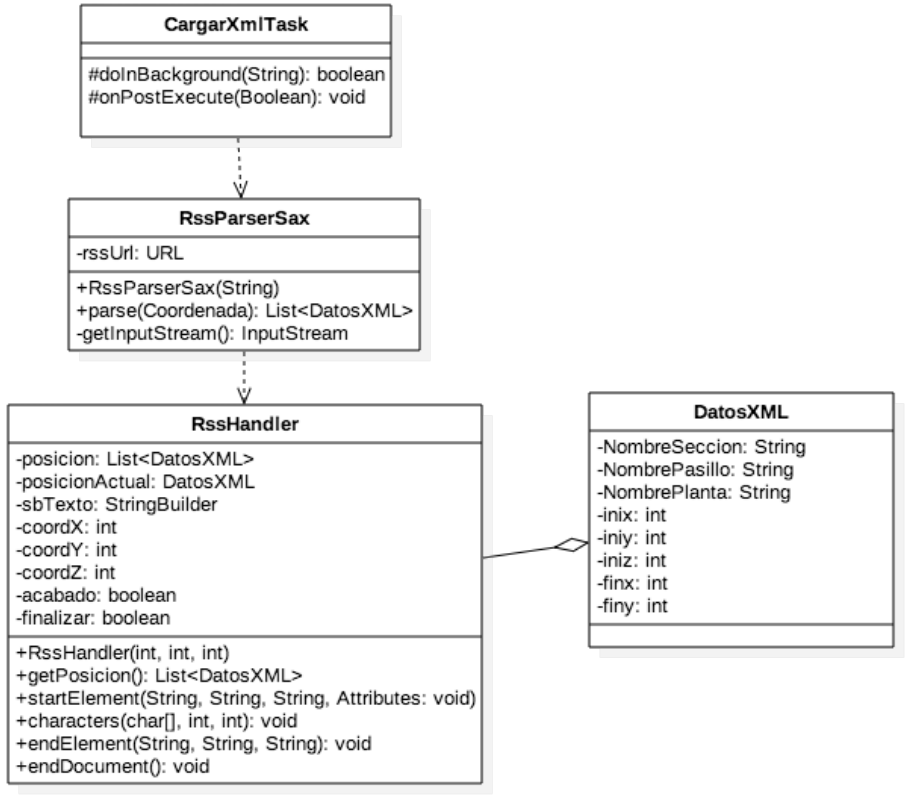


Figura 3.25: Diagrama del parseo y arquitectura *xml*

funcionalidad. En nuestro caso, hemos supuesto que los usuarios querrían, en algún momento, recuperar la historia que realizaron. Por lo tanto, nos vimos en la necesidad de guardar aquellas narrativas que los usuarios quisieran almacenar para ser recuperadas más tarde. Para ello, realizamos una pequeña página web en HTML a la que podemos acceder a través del siguiente link `http://tot.fdi.ucm.es/parable/parableweb/index.php`. La web está almacenada en el servidor proporcionado por la facultad y utilizamos tecnología PHP para insertar las historias desde la aplicación móvil. Concretamente, utilizamos el archivo insertarEnHistoria.php para realizar la inserción en la tabla historia. Este proceso es análogo al visto en la sección de mapeo 3.7. Desde la aplicación, se ofrece la posibilidad de guardar la narración en nuestra base de datos y si se pulsa esta opción, nos ofrecerá darle un nombre (que puede ser el nuestro), se crea un nuevo hilo (*asyntask*) y se ejecuta el servicio *HttpServices* para poder conectarse al servidor y lanzar el archivo PHP descrito antes. Una vez incluida en nuestra base de datos se podrá consultar en cualquier momento introduciendo el nombre que le hayamos dado a la historia.

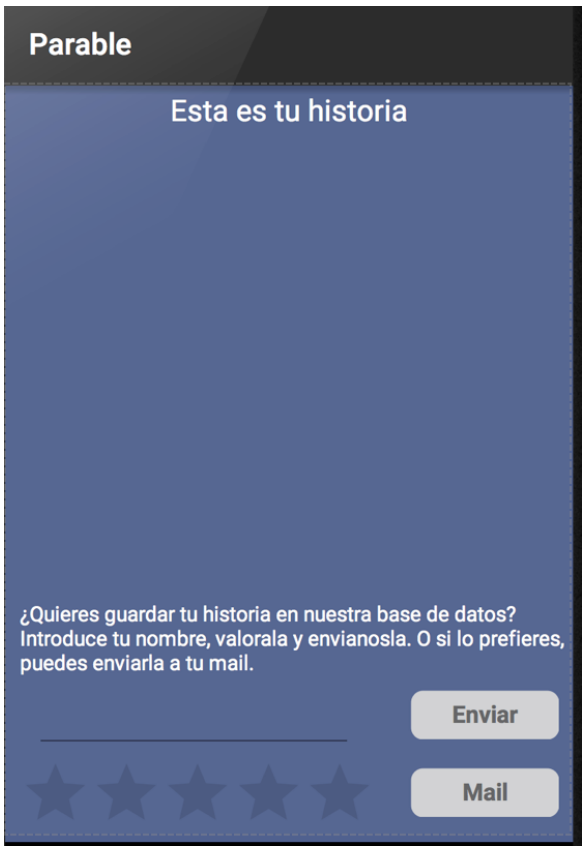


Figura 3.26: Actividad que permite enviar y valorar la historia a la página web

La web consta de cuatro páginas:

- Página de inicio (index.php): donde se muestran las tres últimas historias que se han realizado y guardado en la base de datos.

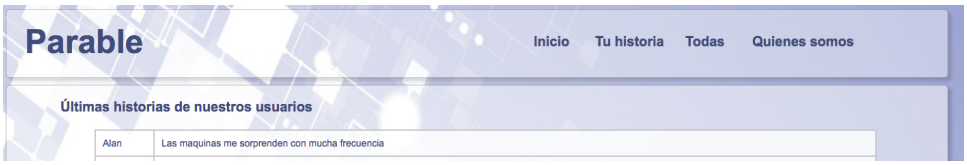


Figura 3.27: Página index.php de la web de PARABLE

- Tu historia (tuhistoria.php): donde se puede consultar, después de introducir el nombre adecuado, la historia que ha realizado una historia concreta.

Figura 3.28: Página tuhistoria.php de la web de PARABLE

- Todas: (todas.php): donde se pueden ver todas las historias de todos los usuarios que han querido almacenar sus narrativas.


Parable
Inicio
Tu historia
Todas
Quienes somos
Todas nuestras historias
Christian
Esta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contarEsta es mi historia y te la voy a contar
Alan2
Las m@quinas me sorprenden con mucha frecuencia.
Luis
La historia de luisLa historia de luisLa historia de luisLa

Capítulo 4

Gestion de proyecto

Si vis pacem, para bellum.
Flavio Vegecio Renato

Resumen: En este capítulo da a conocer la gestión de los riesgos, las especificaciones de casos de uso y los requisitos funcionales y no funcionales que intervienen en el proyecto. También se da una primera aproximación de la planificación del proyecto y sus diferentes iteraciones

4.1. Planificación del proyecto

La planificación del proyecto se ha dividido en cinco iteraciones. Una primera iteración sirvió para elegir las tecnologías más adecuadas para llevar a cabo el proyecto, conocer los entornos de implementación y crear un prototipo para evaluar el alcance del proyecto. A continuación, las siguientes tres iteraciones han servido para el desarrollo del Posicionamiento WiFi y Procesamiento de Lenguaje Natural. Cada una de estas iteraciones se ha subdividido, a su vez, en hitos de 2 a 4 semanas para lograr una flexibilidad dentro de la iteración, y en el proyecto. En la quinta iteración se han realizado las pruebas de campo, mejoras en la eficiencia de la aplicación y adición de funcionalidades menores. Se puede visualizar la duración de estas iteraciones e hitos en los diagramas de Gantt del capítulo 5.

4.2. Gestión de riesgos

4.2.1. Introducción

En este apartado se identifican y analizan los posibles riesgos que pueden ocurrir durante la ejecución de nuestro trabajo. Asimismo, se establece una

priorización de los riesgos y un control y estudio de los mismos. La gestión del riesgo es uno de los elementos clave a la hora de asegurar el éxito en el proyecto, medido en términos de cumplimiento de plazos, costes, alcance funcional y calidad final de la solución. Implantar una gestión del riesgo adecuada será un elemento decisivo a la hora de asegurar que el proyecto se desarrolle correctamente, dentro de los plazos y presupuestos establecidos.

En nuestro proyecto hemos aplicado una estrategia proactiva, que consiste en anticiparse al riesgo para así evitarlo. Por último, usaremos el modelo Boehm(B, 1988) para la gestión de riesgos que establece los siguientes pasos:

1. Identificación del riesgo
2. Análisis del riesgo
3. Priorización del riesgo
4. Control del riesgo

La técnica de priorización que usaremos para determinar cada riesgo es una tabla SQAS-SEI que los relaciona los riesgos por probabilidad y consecuencia. Figuras 4.1 y 4.2

4.2.2. Identificación de riesgos

En este punto se enumeran los distintos riesgos divididos en tres tipos:

1. Riesgos del proyecto:

RP-1: Un miembro abandona el proyecto.
RP-2: Planificación poco realista.
RP-3: Algún miembro no tiene conocimiento de cómo llevar un área.
RP-4: Desconocimiento de las tecnologías.
RP-5: Cambio en los requisitos.
RP-6: Requisitos no identificados.
RP-7: Problemas de compatibilidad del software.
RP-8: Pérdida de interés (desmotivación, baja productividad...).
RP-9: Mala organización y administración del equipo de desarrollo.

2. Riesgos técnicos:

RT-1: El equipo de desarrollo no sigue la planificación.
RT-2: Mal funcionamiento de las aplicaciones que usamos para el proyecto.
RT-3: No tener los medios para desarrollar el proyecto.

Probabilidad	Descripción	Gravedad	Consecuencia
Frecuentes	Ocurre con frecuencia. (Más de una vez al año)	Catastrófico	Aumenta la fecha de entrega en 6 meses. Sobrecoste de 10% Reducción del 10% en funcionalidad
Probable	Ocurre repetidas veces al año. (Frecuencia 1-10^-1)	Crítico	Aumenta la fecha de entrega en no más de 6 meses. Sobrecoste de menos 10% Reducción del 10% de funcionalidad.
Ocasional	Podría ocurrir alguna vez. (Frecuencia 10^-1-10^-2)	Serio	Aumenta la fecha de entrega en no más de 3 meses. Sobrecoste de menos 5% Reducción del 5% de funcionalidad.
Remoto	Poco probable pero posible. (Frecuencia 10^-2-10^-3)	Menor	Aumenta la fecha de entrega en no más de 1 mes. Sobrecoste de menos 2% Reducción del 2% de funcionalidad.
Improbable	Tiene una probabilidad cercana a 0.	Despreciable	Impacto despreciable en el proyecto

Figura 4.1: Probabilidad de los riesgos

Probability / Severity	Frequent	Probable	Occasional	Remote	Improbable
Catastrophic	IN	IN	IN	H	M
Critical	IN	IN	H	M	L
Serious	H	H	M	L	T
Minor	M	M	L	T	T
Negligible	M	L	T	T	T
LEGEND	T = Tolerable	L = Low	M = Medium	H = High	IN = Intolerable

Figura 4.2: Tabla SQAS-SEI

RT-4: Pérdida de información por parte del equipo de desarrollo.
RT-5: Falta de rendimiento en la base de datos.
RT-6: El ratio de defectos solucionados es demasiado lento.
RT-7: El tamaño del software ha sido subestimado.
RT-8: Mala gestión de riesgos.

3. Riesgos del negocio: En este caso, nuestro proyecto no dispone de riesgos del negocio, ya que es una aplicación realizada para un Trabajo de Fin de Grado por lo que es para uso académico.

4.2.3. Análisis de riesgos

El análisis de riesgos determina la probabilidad y la consecuencia asignadas a cada riesgo, mediante el uso de la tabla SQAS-SEI.

RP-1: Un miembro abandona el proyecto	
Descripción	Puede que un miembro deje de realizar las tareas asignadas con el consecuente abandono del proyecto
Probabilidad	Improbable
Gravedad	Crítico
Nivel de riesgo	Bajo

RP-2: Planificación poco realista	
Descripción	No asignar adecuadamente los tiempos que se deberían dar para cada parte del proyecto
Probabilidad	Ocasional
Gravedad	Crítico
Nivel de riesgo	Alto

RP-3: Algún miembro no tiene conocimiento de cómo llevar un área	
Descripción	Puede ser que algún miembro (o varios) del equipo no conozcan como ejecutar una determinada tarea
Probabilidad	Probable
Gravedad	Serio
Nivel de riesgo	Alto

RP-4: Desconocimiento de las tecnologías	
Descripción	Uno o varios miembros del equipo no conocen las tecnologías adecuadas para llevar a cabo una tarea
Probabilidad	Probable
Gravedad	Serio
Nivel de riesgo	Alto

RP-5: Cambio en los requisitos	
Descripción	Alteración en los requisitos que estaban especificados al principio del proyecto
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

RP-6: Requisitos no identificados	
Descripción	No se han identificado todos los requisitos necesarios para el proyecto
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

RP-7: Problemas de compatibilidad del software	
Descripción	Uno de los miembros trabaja sobre el mismo software pero con distintas versiones que provoquen incompatibilidades
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

RP-8: Pérdida de interés (desmotivación, baja productividad...)	
Descripción	La actitud de los miembros del proyecto no es la mejor y el proyecto se ve ralentizado
Probabilidad	Remoto
Gravedad	Serio
Nivel de riesgo	Bajo

RP-9: Mala organización y administración del equipo de desarrollo	
Descripción	Una división del trabajo del proyecto no equitativa puede dar como resultado una descordinación del mismo
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

RT-1: El equipo de desarrollo no sigue la planificación	
Descripción	El equipo no se ajusta a la planificación inicial
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

RT-2: Mal funcionamiento de las aplicaciones	
Descripción	Errores en las aplicaciones o que el proveedor no aporte el servicio necesario para continuar
Probabilidad	Remota
Gravedad	Menor
Nivel de riesgo	Tolerable

RT-3: No tener los medios para desarrollar el proyecto	
Descripción	Puede darse el caso que para llevar a cabo determinada tarea se necesite una tecnología o un equipo específico al cual no se tiene acceso
Probabilidad	Remoto
Gravedad	Menor
Nivel de riesgo	Tolerable

RT-4: Pérdida de información por parte del equipo de desarrollo	
Descripción	Organización errónea de las partes del proyecto que dan como resultado la pérdida de información
Probabilidad	Remoto
Gravedad	Serio
Nivel de riesgo	Bajo

RT-5: Falta de rendimiento en la base de datos	
Descripción	La base de datos elegida no es lo suficientemente potente para llevar a cabo las transacciones
Probabilidad	Remoto
Gravedad	Serio
Nivel de riesgo	Bajo

RT-6: El ratio de defectos solucionados es demasiado lento	
Descripción	El software tiene errores que no se solucionan a tiempo mientras tanto se van encontrando otros nuevos
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

RT-7: El tamaño del software ha sido subestimado	
Descripción	La aplicación ocupa demasiado lugar para incluirse en el dispositivo pensado inicialmente
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

RT-8: Mala gestión de riesgos	
Descripción	Los riesgos se planifican y gestionan de una forma incorrecta que puede incurrir en más riesgos o empeorar el gestionado en ese momento
Probabilidad	Ocasional
Gravedad	Serio
Nivel de riesgo	Medio

4.2.4. Priorización del riesgo

Nivel de Prioridad	Riesgo
Alto	Planificación poco realista
Alto	Algún miembro no tiene conocimiento de cómo llevar un área
Alto	Desconocimiento de las tecnologías
Medio	Cambio en los requisitos
Medio	Requisitos no identificados
Medio	Problemas de compatibilidad del software
Medio	Mala organización y administración del equipo de desarrollo
Medio	El equipo de desarrollo no sigue la planificación
Medio	El ratio de defectos solucionados es demasiado lento
Medio	El tamaño del software ha sido subestimado
Medio	Mala gestión de riesgos
Bajo	Un miembro abandona el proyecto
Bajo	Pérdida de interés (desmotivación, baja productividad...)
Bajo	Pérdida de información por parte del equipo de desarrollo
Bajo	Falta de rendimiento en la base de datos
Tolerable	Mal funcionamiento de las aplicaciones
Tolerable	No tener los medios para desarrollar el proyecto

4.2.5. Control de riesgos

La información esencial sobre el proceso de gestión del riesgo lo vamos a incluir en un Plan de Reducción, Supervisión y Gestión del Riesgo (RSGR). En nuestro proyecto vamos a elaborar el plan de control de riesgos sobre aquellos que tengan un impacto alto o intolerable ya que son los tendrían un impacto que podría retrasar la entrega de proyecto.

Planificación poco realista	
Plan de Reducción	Adecuar bien los tiempos antes de empezar a realizar las tareas al número de miembros del equipo y del tiempo disponible
Plan de Supervisión	Revisar en cada reunión semanal como avanza el proyecto y las diferentes tareas asignadas a cada miembro
Plan de Gestión	Reasignación de personal a las tareas que causan que no se cumpla la planificación o aumentar el tiempo de horas invertidas en ellas

Algún miembro no tiene conocimiento de cómo llevar un área	
Plan de Reducción	Establecer durante la planificación una serie de herramientas que sean similares a algunas ya usadas durante el curso
Plan de Supervisión	Asegurar que se utilizan las herramientas que se han predefinido en la planificación y que se hace adecuadamente
Plan de Gestión	Buscar herramientas parecidas o que hagan la misma función que aquella que no se sabe utilizar

Desconocimiento de las tecnologías	
Plan de Reducción	Asignación del trabajo de forma que cada miembro domine la tecnología que tiene que usar
Plan de Supervisión	Asegurar que cada miembro del grupo domine la tecnología correspondiente
Plan de Gestión	Buscar otras tecnologías iguales o parecidas de forma que puedan llegar a conocerla los miembros del grupo

4.3. Análisis de requisitos y especificación de casos de uso

En esta sección explicaremos los requisitos del proyecto. Distinguimos entre requisitos funcionales y no funcionales. Los primeros son aquellos servicios que proporciona el sistema (funciones), las respuestas del sistema ante determinadas entradas y el comportamiento del sistema en situaciones particulares. En nuestro caso, que la aplicación nos posicione, que haga uso de una capa de Realidad Aumentada, etc. Los requisitos no funcionales que son las restricciones de los servicios o funciones que ofrece el sistema. Es decir, aquellas necesidades de infraestructura que no realizan ninguna función en la aplicación. Pueden ser, por ejemplo, el lenguaje de codificación, en nuestro caso Java, las necesidades hardware que se necesitan para ejecutarla, etc.

4.3.1. Requisitos funcionales

Son aquellos requisitos que dictan lo que hace el sistema. Son descritos como un conjunto de entradas y precondiciones que dan como resultado una determinada salida y postcondición después de un proceso determinado. En nuestro caso, los principales requisitos funcionales fueron descritos por los directores del proyecto y una vez cumplidos, añadimos los nuestros para completar la funcionalidad de la aplicación.

El siguiente requisito nos muestra la necesidad de implementar una funcionalidad que recupere las WiFi's que hay a nuestro alrededor. Para ello, al entrar en la aplicación, deberemos pulsar en el menú el botón de *Guardar coordenadas* lo que dará como resultado una lista de las redes disponibles. Mostrará el nombre de la red, su dirección MAC y la intensidad.

Escaneo de redes Wifi	
Cod Requisito	RF-1
Prioridad	Alta
Estabilidad	Alta
Descripción	Recoge las MAC's y las intensidades de las redes WiFi que hay alrededor del terminal
Precondición	WiFi activado en el móvil
Postcondición	Se muestran las redes con sus datos
Datos de entrada	Ninguno
Datos de salida	Nombres MAC e intensidades de las WiFi's que rodean al móvil
Flujo de evento	1. Pulsamos en *Guardar Coordenadas* 2. Se muestran en la interfaz las redes
Excepciones	1. No hay o no se encuentran redes WiFi a nuestro alrededor. 2. WiFi desactivado en el móvil

El siguiente requisito nos define la necesidad de guardar en la base de datos las coordenadas encontradas por el dispositivo. Para ello deberemos apoyarnos en el requisito RF-1 y a continuación seleccionaremos las redes que queremos guardar, introduciremos las coordenadas que les pertenecen a esas determinadas intensidades y pulsamos *Enviar*. A continuación todas las redes, intensidades y las coordenadas son insertadas en la base de datos.

Insertar coordenadas en la base de datos	
Cod Requisito	RF-1.1
Prioridad	Alta
Estabilidad	Alta
Descripción	Guarda en la base de datos las MAC's, las intensidades y las coordenadas X, Y, Z que le asignemos
Precondición	RF-1 (Escaneo de redes WiFi)
Postcondición	Se han guardado correctamente los datos en la base de datos
Datos de entrada	MAC's, intensidades, X, Y y Z
Datos de salida	Ninguno
Flujo de evento	1. RF-1 2. Pulsamos en cualquiera de las redes 3. Introducimos las coordenadas y le damos a enviar
Excepciones	1. No hay conexión con la base de datos.

El siguiente requisito se encarga de posicionar al usuario. Con la aplicación ejecutada, el WiFi del dispositivo activo y el mapeo del lugar donde queremos posicionarnos pulsamos la opción de *Comenzar relato*. Al entrar, automáticamente, debe calcular la posición del usuario a través de los diferentes algoritmos.

Posicionar al usuario	
Cod Requisito	RF-2
Prioridad	Alta
Estabilidad	Alta
Descripción	Nos dice en lenguaje natural donde estamos actualmente.
Precondición	Tener la base de datos con las coordenadas de la zona en la que nos encontremos. WiFi activado
Postcondición	Muestra en lenguaje natural donde nos encontramos
Datos de entrada	Ninguno
Datos de salida	Situación actual dentro de la facultad
Flujo de evento	1. Pulsamos la opción *Comenzar relato* 2. Calcula la posición y la muestra por pantalla en lenguaje natural
Excepciones	1. No nos encontremos en un lugar donde no hayamos guardado las coordenadas e intensidades

Este requisito define la funcionalidad de leer un QR. Una vez que estamos ejecutando la aplicación y pulsamos *Comenzar relato* podremos en cualquier momento pulsar el botón de *Leer QR* donde se recogerán los puntos y la información que contiene para añadirlo a la narrativa.

Leer Código QR	
Cod Requisito	RF-3
Prioridad	Alta
Estabilidad	Alta
Descripción	La aplicación recoge el código QR y muestra la información correspondiente
Precondición	Cámara activa
Postcondición	Muestra la información correspondiente al código.
Datos de entrada	Código QR.
Datos de salida	Información relacionada con el código.
Flujo de evento	1. Pulsamos la opción *Leer QR*. 2. Busca en el XML la información correspondiente. 3. Añade la información del QR.
Excepciones	1. No se lee bien el código. 2. Leemos un código que no contiene información en nuestro XML.

El requisito de mostrar narración es la funcionalidad principal de PARABLE. Ella mostrará la narración de la actividad del usuario así como los puntos obtenidos y el audio en el caso de tratarse de puntos de interés.

Mostrar narración	
Cod Requisito	RF-4
Prioridad	Alta
Estabilidad	Alta
Descripción	Narra la situación actual del usuario
Precondición	RF-3
Postcondición	Muestra la narración correspondiente a la situación del usuario.
Datos de entrada	Ninguno
Datos de salida	Información narrativa.
Flujo de evento	1. RF-3 nos devuelve la situación del usuario. 2. Se buscan frases alternativas en la base de datos. 3. Se elige alguno. 4. Se busca la plantilla narrativa adecuada y se rellenan los huecos. 5. Se muestra por pantalla la plantilla rellena.
Excepciones	1. Ninguna.

Este requisito muestra toda la narración surgida de la actividad del usuario una vez que haya acabado de usarla y pulse *Ver historia completa*.

Mostar la narración completa	
Cod Requisito	RF-5
Prioridad	Media
Estabilidad	Alta
Descripción	Se muestra toda la narración en la que ha participado el usuario por pantalla.
Precondición	Ninguna
Postcondición	Ninguna
Datos de entrada	Plantillas con los huecos rellenos.
Datos de salida	El texto completo
Flujo de evento	1. Pulsamos la opción *Ver historia*, 2. Concatena las historias y las muestra.
Excepciones	1. Ninguna.

La aplicación deberá poder enviar la historia a un determinado mail o tweetear el paso que desee el usuario.

Envio al email o Twitter de la historia	
Cod Requisito	RF-6
Prioridad	Media
Estabilidad	Alta
Descripción	Se envía toda la narración al mail o Twitter.
Precondición	Haber participado en una narración.
Postcondición	Ninguna
Datos de entrada	Narración completa y un mail.
Datos de salida	Ninguno.
Flujo de evento	1. Pulsamos la opción *Enviar al Mail*, 2. Introducimos el mail 3. Pulsamos *Enviar*
Excepciones	1. Ninguna.

4.3.2. Requisitos no funcionales

Son aquellos requisitos que no intervienen en la funcionalidad de la aplicación sino en aquello que necesita la aplicación para ejecutarse.

4.3.2.1. Software

- Versión de Java 1.8: es la versión de Java que usaremos para desarrollar la aplicación.
- Versión de Android 5.0 (Lollipop)
- Librerías de Realidad Aumentada Vuforia. Requisito para poder llevar a cabo la capa de Realidad Aumentada superpuesta sobre la vista de cámara.
- MySQL 5.0.67
- PHP 5.0.0 para funciones más seguras como MYSQLI.

4.3.2.2. Hardware

Debido a que la aplicación está pensada para la última versión de Android 5.0 (Lollipop) los requisitos del teléfono móvil deben ser los siguientes:

- Terminal Móvil que pueda usar Android
- Procesador de 64 bits
- 1 Gb de memoria RAM
- 8 Gb de memoria suficientes para contener el sistema operativo y nuestra aplicación.
- Cámara fotográfica para realizar la captura de los códigos QR

Samsumg Galaxy Note 3: Debido a los requisitos no funcionales relacionados con el hardware hemos optado por un este móvil cedido por el grupo de investigación NIL.

Servidor Apache: Servidor para gestionar la base de datos a la que accederá la aplicación para recuperar los datos.

Capítulo 5

Desarrollo del proyecto

Para mi he dejado lo mejor: la esperanza
Alejandro Magno

Resumen: Este capítulo se describirán las distintas iteraciones y como se han llevado a cabo mostrando la planificación mediante diagramas gantt.

5.1. Iteraciones

El proyecto ha sido dividido en cinco iteraciones.

5.1.1. Primera iteración: Prototipo

La primera iteración consistió en una toma de contacto con las distintas tecnologías que íbamos a utilizar. También realizamos un pequeño prototipo de pruebas para conocer en profundidad Android y la viabilidad de la aplicación. Para terminar, organizamos el proyecto y planificamos los tiempos del mismo.

5.1.2. Segunda iteración: Posicionamiento WiFi

En esta segunda iteración nos centramos en el posicionamiento WiFi. Probamos distintas soluciones para conseguir un posicionamiento *indoor* adecuado y fiable. En esta etapa realizamos ya una pequeña interfaz para poder introducir coordenadas en la base de datos y pequeños prototipos de posibles funcionalidades futuras. Al final de la iteración comenzamos a mapear la facultad y pruebas de posicionamiento básicas.

5.1.3. Tercera iteración: Generación de Lenguaje Natural

En la tercera etapa nuestro objetivo se centro en la generación de lenguaje natural, construir el mapa de la facultad mediante xml y realizar algunas pruebas de posicionamiento más avanzadas con las que conseguimos ya que nos mostrase nuestra situación mediante lenguaje natural. Se terminó la etapa mapeando otra sección de la facultad.

5.1.4. Cuarta iteración: Código QR, mapeo e interfaces

En la cuarta iteración introdujimos la funcionalidad de leer códigos QR para el dinamismo de la aplicación. Desarrollamos la facultad en forma de grafo, apoyados en los objetivos de la tercera iteración para poder crear caminos entre las secciones y facilitar el seguimiento de la historia. En esta etapa, también se mejoró la narrativa y se terminó de mapear el primer piso de la facultad y se realizaron pruebas de campo para comprobar funcionamiento y eliminar *bugs*. Por último, se realizó una interfaz más amigable en toda la aplicación.

5.1.5. Quinta iteración: Eficiencia, pruebas y otras funcionalidades

En esta última etapa se mejoró la eficiencia de la aplicación. Además, se añadieron las funcionalidades de crear nuestros propios códigos QR, poder *twittear* nuestra situación actual, enviar la narrativa completa por mail o a una página web que también fue desarrollada por nosotros. En esta iteración también se añadió la tecnología *text to speech* para poder dotar de audio o lenguaje hablado a la aplicación. Para terminar, se realizaron las pruebas de campo finales y eliminación de errores.

5.2. Proceso de trabajo

Esta sección contiene información más detallada de todo el proceso de desarrollo del proyecto, según la metodología Pressman (2005).

5.2.1. Fase inicial

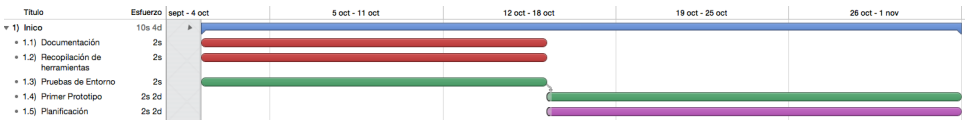


Figura 5.1: Etapa de inicio.

En la fase de inicio tomamos contacto con las diferentes tecnologías, investigamos sobre ellas y realizamos pruebas para familiarizarnos con el entorno. A continuación realizamos un primer prototipo de la aplicación y planificamos el trabajo a realizar durante todo el proceso.

5.2.2. Fase de elaboracion

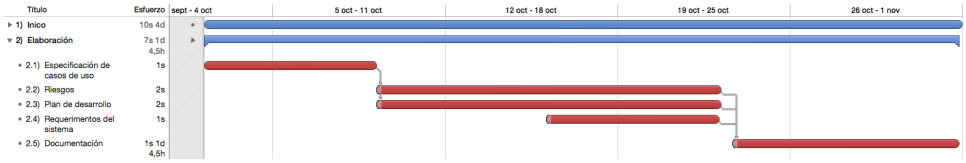


Figura 5.2: Etapa de elaboración.

En esta etapa elaboramos los requisitos funcionales y no funcionales de PARABLE y realizamos un estudio sobre los riesgos del proyecto, como vamos a planificar el desarrollo y comenzamos con un primer contacto sobre la documentación del proyecto. Es decir, definimos formalmente el proyecto. Aunque la mayor parte de esta etapa permaneció sin cambios, el intercambio de opiniones y *feedback* de nuestros directores dio lugar a pequeños cambios.

5.2.3. Fase de construccion

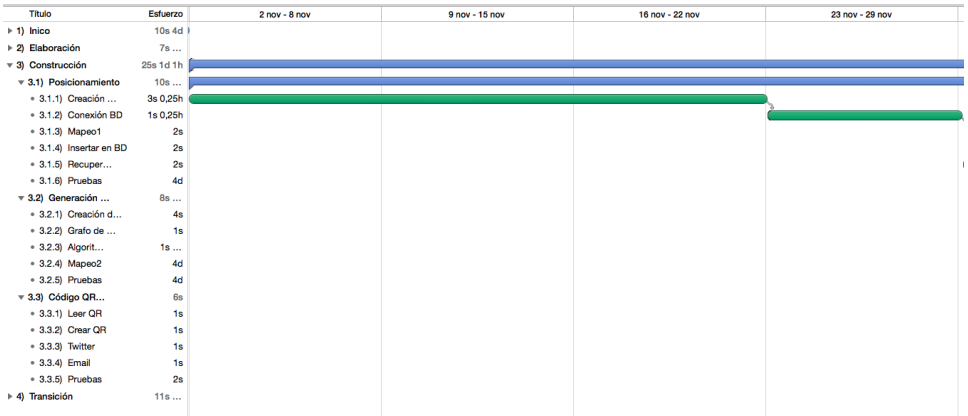


Figura 5.3: Primera etapa de construcción.

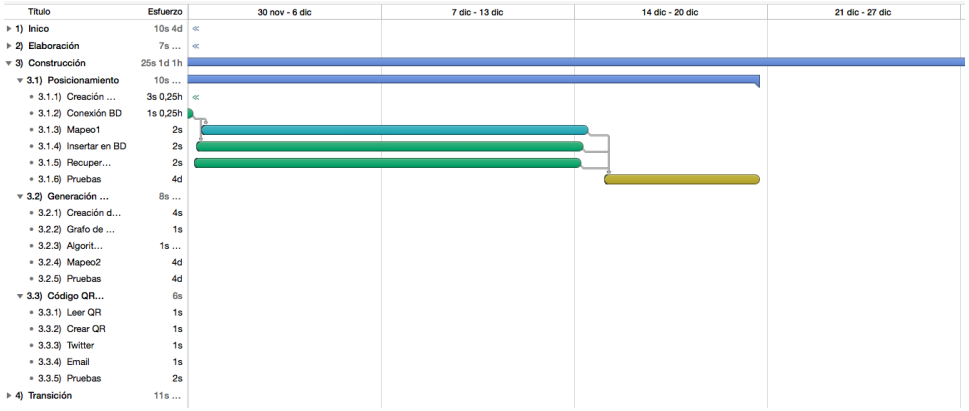


Figura 5.4: Segunda etapa de construcción.

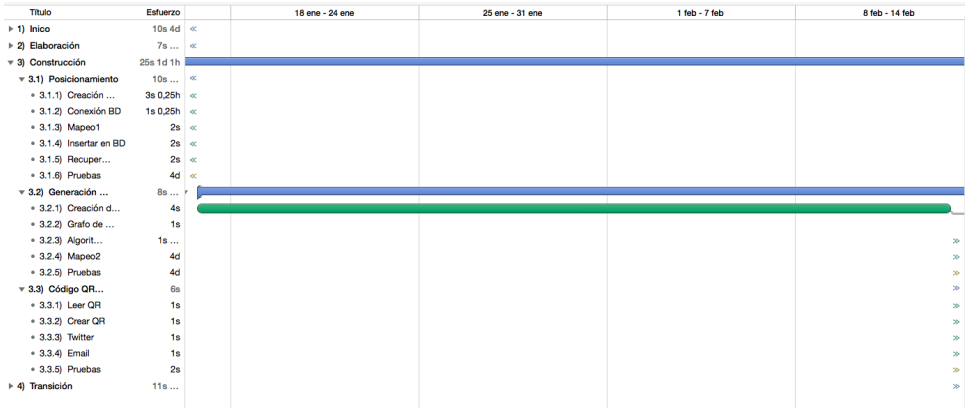


Figura 5.5: Tercera etapa de construcción.

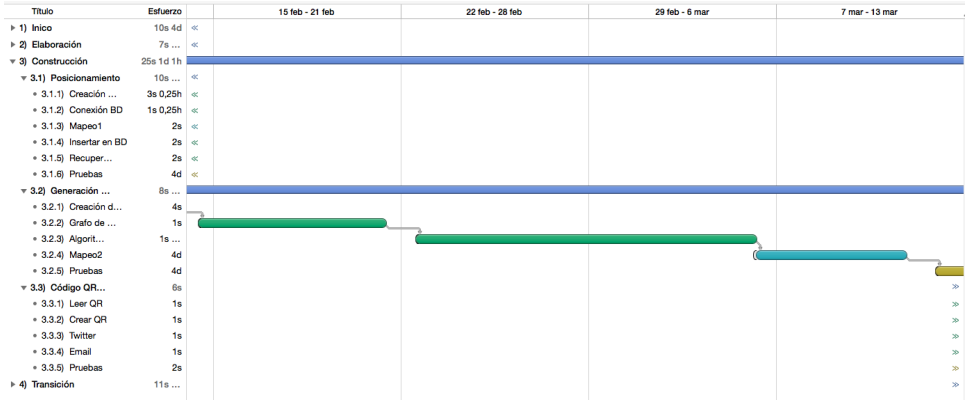


Figura 5.6: Cuarta etapa de construcción.

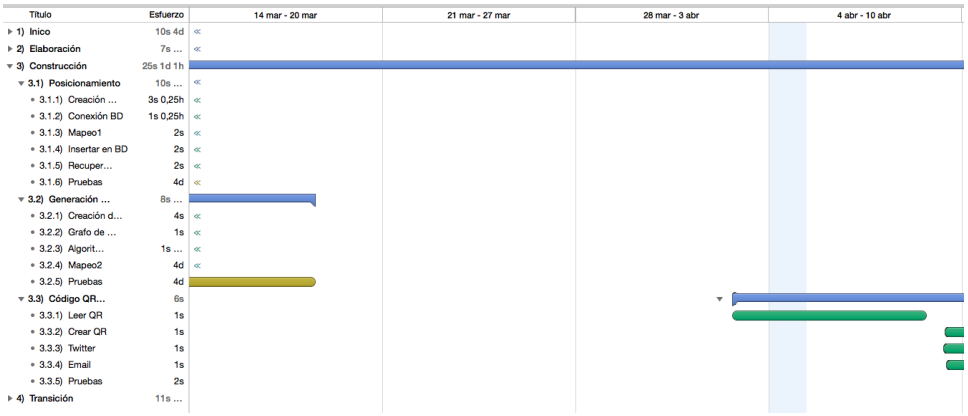


Figura 5.7: Quinta etapa de construcción.

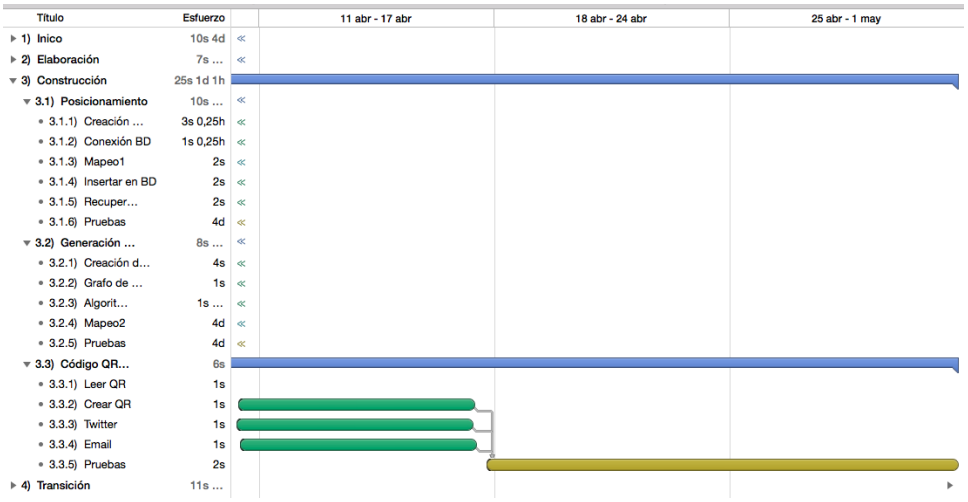


Figura 5.8: Sexta etapa de construcción.

La fase de construcción se ha dividido en varias partes debido a la longitud del diagrama y para facilitar su comprensión. Como se ha explicado en en el punto 5.1, la fase principal de desarrollo tiene tres iteraciones; una para cada uno de los puntos principales de la aplicación. La iteración del posicionamiento WiFi es la más larga porque usamos parte del algoritmo desarrollado por Enrique López Mañas (2010). Al usar código desconocido nos tuvimos que familiarizar con el así como comprender el funcionamiento en profundidad para poder adaptarlo a nuestras necesidades. De igual manera, aquellas partes que no necesitábamos tenían que ser eliminadas sin perder eficiencia y eficacia, y añadir de la misma manera, funciones que nos hacían falta. Aunque no esté reflejado en los diagramas, esta fase de construcción tiene una gran carga de documentación debido a la necesidad de buscar, usar, conocer y aplicar librerías externas.

5.2.4. Fase de transición

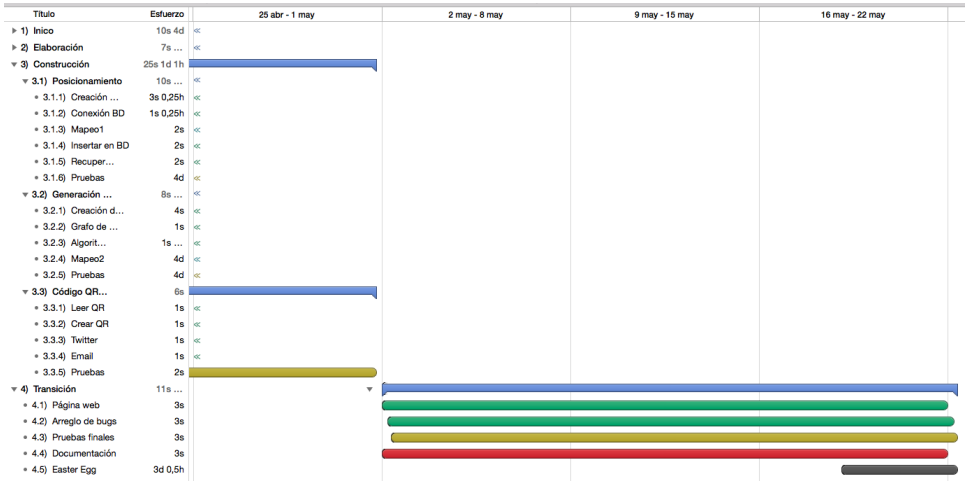


Figura 5.9: Etapa de transición.

En la última etapa, transición, se ha mejorado la eficiencia y se han añadido funcionalidades secundarias para completar la aplicación. Se hicieron las pruebas finales y el arreglo de *bugs*. También se creo un sitio *web* para poder también ver las narrativas desde el navegador y para finalizar se realizó toda la documentación y manual de usuario.

Capítulo 6

Conclusiones y trabajo futuro

Hola, me llamo Íñigo Montoya. Tú mataste a mi padre, prepárate a morir.

Íñigo Montoya (Mandy Patinkin)

Resumen: Este capítulo presenta las conclusiones, posibles aplicaciones futuras del proyecto y el trabajo realizado por cada alumno

6.1. Conclusiones

La solución propuesta para la generación de historias centradas en el usuario, usando plataformas móviles y un servidor central con la información, ofrece una visión de como se puede utilizar el posicionamiento WiFi en entornos *indoor* para narrar el uso de las instalaciones y el recorrido de las mismas. Usando esta localización, se puede generar un *log* de todos los pasos seguidos por los usuarios automatizando, llegado el caso, el análisis de los datos. Por ejemplo, de las áreas más usadas o los recorridos típicos y como mejorar el uso de las instalaciones. Apoyados en los códigos QR, también podremos comprobar cuales son los objetos más usados y ofrecerlos de una forma más accesible a los usuarios mejorando la experiencia que se hace de los mismo. Por último también sería posible detectar si algún usuario accede a zonas prohibidas para él y así lo veríamos reflejado en el log antes propuesto.

Al finalizar PARABLE, la conclusión a la que hemos llegado es que recuperar la posición mediante WPS requiere un alto grado de computación que se ve reflejado en el consumo de recursos y el consiguiente gasto de batería del dispositivo móvil. Aunque elegimos esta tecnología, apoyados en el proyecto AVANTI de años anterior, creemos que las tecnologías han avanzado lo

suficiente en estos años para que sea más factible usar, por ejemplo, *beacons bluetooth*. Estos *beacons*, a diferencia del WPS que cada pocos centímetros tiene que recalcular la posición, solo hay que introducirles la información de la posición una vez por cada uno. Por ello, el mapeo del edificio en el que se ejecute la aplicación es menos tedioso.

Si nos centramos en el uso de Android como plataforma, creemos que fue la acertada debido a la cantidad de documentación y código *open source* disponible en la red. También dispusimos de una gran cantidad de librerías que nos ayudaron a la hora de realizar el proyecto, como es el caso de *jGraph* para transformar la facultad en un grafo.

Por lo tanto, después de realizar el presente proyecto, podemos decir que es factible crear una narrativa más o menos compleja, apoyándonos en el posicionamiento WiFi. Las posibilidades a partir de aquí son infinitas. Como se ha comentado anteriormente, este desarrollo podría servir para el seguimiento de personas dentro de una empresa y así, aumentar la seguridad. Para personas con problemas de memoria y se les puede hacer un seguimiento dentro de un edificio o en sus propias casas y que, al leer de nuevo su historia les ayude a recordad. También se podría aprovechar para recuperar el movimiento de las personas de un edificio para ver como se pueden optimizar los tiempos y mejorar la organización del propio edificio a partir de conocer las narraciones de las propias historias. Para terminar, también se podría aplicar a algún juego en el que haya que moverse por localizaciones indoor con objetivos y que los demás usuarios puedan leer tus movimientos haciéndoles, de esta manera, más partícipes en la misma.

6.2. Conclusions

The proposal for the creation of stories focusing on the user, using mobile platforms and a central server with our information, we offer a vision of how you can use the WiFi positioning in indoor environments to narrate the use of the facilities and the route of the same. Using this location, you can generate a log of all steps taken by users, if necessary, the analysis of data. For example, the most used areas or typical routes and how to improve the use of the facilities. Supported by QR codes, we also see which are the most used objects and offer a more accessible way to improve the users experience is the same. Finally it would also be possible to detect whether a user accesses areas prohibited for him and so we would see reflected in the log before proposed.

At the end, PARABLE, the conclusion we have reached is to regain position by WPS requires a high degree of computation that is reflected in the consumption of resources and the consequent cost of mobile device battery. Although we chose this technology, supported in previous years AVANTI project, we believe that technologies have advanced enough in recent years

to make it more feasible to use, for example, *Bluetooth beacons*. These *beacons*, unlike the WPS every few centimeters have to recalculate the position, you just have to introduce the information once for each position. Therefore, the mapping of the building in which the application runs is less tedious.

If we focus on the use of Android as a platform, we believe it was successful because of the amount of documentation and code *open source* available online. Also we arranged a lot of libraries that helped us when carrying out the project, as is the case of *Jgraph* to transform the power in a graph.

Therefore, after developing this project, we can say that it is feasible to create a more or less complex narrative, relying on WiFi positioning. The possibilities from here are endless. As discussed above, this development could be used to track people within a company and thus increase security. For people with memory problems and they can track inside a building or in their own homes and that in reading your story again help them remember. It could also take the opportunity to regain movement of people from a building to see how they can optimize time and improve the organization of the building from knowing the stories of the stories themselves. Finally, it could also be applied to a game in which you have to move for indoor locations with targets and that other users can read your making movements, thus more involved in it.

6.3. Trabajo futuro

Aunque PARABLE se ha centrado en la generación de historias a partir de la posición del usuario dentro del edificio, el conseguir llevar a cabo este modelo ofrece nuevas posibilidades. Una de ellas es la de poder ser utilizada por personas con problemas de memorias como podría ser el Alzheimer y que narren el día a día de la persona para que pueda ejercitar la memoria a partir de releer los movimientos que ha llevado a cabo. Otro de las aplicaciones futuras, podría ser el control de los usuarios dentro de un edificio y las zonas prohibidas permitidas de los mismos con el consiguiente control de seguridad. Por último, un objetivo más lúdico, sería el de una yincana en el que se narrase toda la aventura y las consiguientes pruebas.

En el futuro, los próximos pasos de PARABLE podrían ser:

- Incorporar una capa de Realidad Aumentada que muestre de una forma más intuitiva los elementos que pueden ser usados en el entorno.
- Mejorar la usabilidad para facilitar el uso a las personas con discapacidades.
- Mejorar el tiempo de procesamiento de datos.
- Aumentar el número de elementos usables en el entorno y la forma de interactuar con ellos más allá de leer un código QR.

- Mejorar el sistema narrativo dando mayor fluidez y complejidad a la existente.
- Ofrecer diferentes lenguajes a la hora de narrar la historia.
- Visionado

6.4. Trabajo realizado por cada miembro

En esta sección se enumerará el trabajo realizado por cada uno de los miembros del proyecto. Se muestra tanto la realización de la memoria, como el desarrollo del código así como las pruebas de campo.

6.4.1. Christian Álvarez

Al inicio del proyecto nos encontramos con el principal problema de no conocer el entorno y no haber programado nunca sobre *Android*. Esta fue una de las primeras tareas que realicé, familiarizarme con el compilador Android Studio. Una vez hecho esto, comprobé que todo el entorno, formado por el terminal móvil, el servidor *Tot*, y la base de datos se comunicaban satisfactoriamente. A continuación, comencé a estudiar el algoritmo de posicionamiento de AVANTI. Esta tarea fue bastante complicada y larga, ya que al ser código ajeno, tuve que ir *debuggeando* poco a poco para comprender como recogían las coordenadas, las guardaban y las recuperaban, así como los algoritmos usados para calcular la posición, en este caso el de los vecinos más cercanos. Durante este proceso también realicé pequeñas aplicaciones de las funcionalidades que más tarde usaríamos. Esto fue muy útil ya que permitió conocer mucho más en profundidad *Android* y aquellas peculiaridades que solo se aprenden ensayando.

Una vez que el algoritmo funcionó, mapeamos el primer piso de la facultad con el fin de comprobar sobre el terreno si el algoritmo funcionaba correctamente o solo teóricamente. En este proceso, Kevin iba guardando las intensidades y coordenadas mediante el móvil y yo comprobaba en la base de datos que todo estaba correcto. Esta tarea nos llevo varios días por la cantidad de coordenadas y WiFi's que encontrábamos. Se puede leer más específicamente el proceso en la sección 3.7. La comunicación a través de hilos con el servidor y la base de datos con archivos PHP fue desarrollada también enteramente por mi. Esta parte del proyecto fue problemática debido a las especificaciones de *Android*, ya que este no permite que se bloquee el hilo principal. Así que para la comunicación con el servidor, a través de PHP, hubo que crear nuevos hilos con la sentencias *Asyntask*. Para llegar a esta conclusión hubo que hacer una pequeña tarea de investigación ya que no avisaba de este problema en concreto.

Es importante señalar que durante toda la realización de PARABLE, nos tuvimos que turnar el móvil ya que solo disponíamos de uno con la consecuente ralentización de la aplicación debido a que los emuladores ofrecidos por Android no disponían de receptores WiFi, y no se podía probar la aplicación como era debido. Después de mapear la facultad, el móvil lo tuvo Kevin, así que yo me dediqué a comenzar con la memoria. Durante los primeros meses de proyecto escribí el capítulo del estado del arte y empecé con los requisitos y la planificación del proyecto.

En febrero, volví a tener el móvil y comencé a darle una imagen visual más atractiva. Durante este mes construí toda la interfaz de la aplicación y la portada del mismo. Esta parte, técnicamente, no fue ningún reto más que el de hacerlo atractivo para el usuario. Para este fin, contamos con la ayuda de una estudiante de máster en diseño llamada Ángela Brea Fernández que nos ayudo con la elección de colores, el logo y la realización de la portada. A continuación elaboré la parte de las redes sociales, el envió por mail y el *tweetbot*. Para esta tarea, me apoyé en la API que ofrece *Twitter* que es fácil de usar y muy intuitiva. Para el mail usé un hilo externo para luego apoyarme en la aplicación nativa de *Android* para enviar el mail a quien queramos. Durante el tiempo que me correspondía el móvil también realicé por completo la página de PARABLE, en la que se pueden ver las narrativas si el usuario quiere guardarlas en nuestra base de datos. Todo el sistema fue creado por mi; desde las diferentes funciones PHP (mostrar todas las historias, mostrar la de la propia persona y mostrar las últimas tres narrativas), la página web (HTML, jQuery) y la interfaz que permite, desde el móvil, acceder a estas funcionalidades. Para continuar, en esta etapa construí la funcionalidad de lectura y creación de códigos QR. Para realizar esto, me apoye en la librería *ZXing*. El uso de esta librería también me creó problemas ya que para su uso había que incluir ciertas sentencias en el archivo llamado *Gradle* y este archivo es crítico para el funcionamiento de las aplicaciones con lo que tuve que estar bastante seguro antes de compilar PARABLE porque cabía la posibilidad de provocar errores de difícil solución. Afortunadamente, este no fue el caso y todo siguió el curso correcto. Por último, programé el cálculo del camino más rápido y la comprobación del camino elegido por el usuario (si hacía caso a la narración o no) apoyándome en la librería *jGraph* que me ayudo a gestionar la facultad como si de un grafo se tratase.

Una vez llegó abril, el móvil pasó a manos de Kevin así que yo me centré en escribir la memoria de PARABLE. Durante estos dos meses restantes, hasta junio, escribí lo que quedaba de memoria. La introducción y su traducción completa; el capítulo tres, que contiene toda la información técnica del proyecto; el capítulo cuatro (salvo los riesgos), que contiene la gestión de PARABLE a lo largo de todo el año; el capítulo cinco, que contiene el

desarrollo del proyecto y por último las conclusiones y su traducción.

Durante todo el año realizamos conjuntamente el mapeo y las pruebas de campo, así como la asistencia a las reuniones con los directores del proyecto. Además, las líneas a seguir del proyecto fueron tomadas de forma consensuada.

6.4.2. Kevin Arboleda

Durante la primera parte del trabajo de fin de grado, Christian se quedó con el teléfono y tuvo que encargarse de toda la parte del posicionamiento debido a la dificultad de compaginar mi trabajo con las clases, ya que disponía de mucho menos tiempo. Una vez pude organizar mejor mi trabajo, me encargué de mapear los pasillos del primer piso, mientras Christian comprobaba que todas las coordenadas se habían guardado satisfactoriamente.

Una vez insertadas todas las coordenadas en la base de datos, me dispuse a su traducción creando un *XML* con el que podía estructurarlas dividiéndolas en plantas, pasillos, aulas y secciones. Para ello tuve que investigar su uso, tanto su parseo como la recogida de datos ya que eran totalmente desconocidas para mí.

A continuación, modifiqué toda la aplicación de forma que usaran todos los patrones mencionados en el punto 3.9.1, consiguiendo así mejorar la aplicación desde el punto de vista estructural y de eficiencia. Todo el posicionamiento lo modifiqué, pasando así de que nos mostrara simples coordenadas, a que cuando te posicionas a lo largo de la facultad, lanzara eventos dependiendo de si entrabas o salías de una sección, pasillo o aula, o si por el contrario te quedabas quieto.

Después nos metimos de lleno en la parte de la Generación de Historias, para lo cual, cree un archivo *properties*, en el que se encuentran las frases que usa la aplicación a la hora de contarte lo que estas haciendo. Una vez creado, lo integré con los eventos de forma que dependiendo del evento que se lanzara cogería una frase u otra. Vimos que las frases eran siempre las mismas y daba una sensación poco realista por lo que añadí un array con sinónimos para cambiar distintas palabras de cada frase, consiguiendo así mayor fluidez.

Después de esto, Christian se quedo con el móvil, por lo que me encargué de toda la parte de gestión de riesgos de nuestro proyecto incluyéndolo en la memoria. Cuando volví a tener el móvil y con el cálculo del camino más rápido hecho, me dispuse a realizar una adaptación de este cálculo de forma que el usuario pudiera saber en todo momento qué camino coger. Además añadí una serie de checkpoints por los que el usuario necesitaba pasar para poder avanzar en la historia. También realicé, mediante una serie de funciones, una forma de decirle al usuario cuando ha elegido un camino marcado

por nosotros o cuando ha decidido tomar su propio camino.

Una vez hecho esto me centre en los QR, ya que necesitábamos integrarlos en nuestra aplicación de forma efectiva. Con la función que permitía leer y crear QR realizada por Christian, lo que hice fue crear una tabla en la base de datos que nos permitía guardar los creados por nosotros así como un campo que te decía si esos QR ya habían sido leído o no. Se me ocurrió la idea de que podían estar escondidos por la facultad y ser puntos para el jugador, de forma que modifiqué la interfaz añadiendo un marcador que se incrementaba cada vez que el jugador leía un QR valido. Nos dimos cuenta de que no podían introducirse en la base de datos sin más, por lo que se me ocurrió encriptarlos usando la función MD5 que te ofrece PHP, de esta forma todos los QR que nosotros teníamos en la base de datos están encriptados y los QR generados por el propio usuario, los guardo de la misma forma. Por lo que cuando el usuario se dispone a leer un QR, lo recojo y lo comparo con los guardados en la base de datos, comprobando también si ya están usados.

Después de esto, añadí la función *TextToSpeech* que permitía que el móvil te fuera contando lo que salía por pantalla, consiguiendo así que la aplicación fuera más amena. También realicé por completo la pantalla de valoración con su correspondiente base de datos, de forma que el jugador podía valorar nuestra aplicación, incluyendo una reseña si así lo quería, para ello tuve que modificar parte de la pagina web para que apareciera la valoración así como la creación de los correspondientes PHP para la conexión con la misma.

Para terminar, debo añadir que ha sido difícil avanzar con toda la velocidad que nos hubiera gustado en el proyecto, debido a la limitación de tener solo un móvil y que las pruebas las teníamos que hacer siempre en la facultad. Todos las decisiones han sido tomadas de forma conjunta por lo que se ha podido realizar una buena gestión del proyecto.

Apéndice A

Manual de usuario

Pones tu pie en el camino y si no cuidas tus pasos, nunca sabes a donde te pueden llevar.

John Ronald Reuel Tolkien, El Señor de los Anillos

Resumen: Este apéndice explica detalladamente como usar la aplicación por parte del usuario así como todas las posibilidades de ejecución que nos ofrece.

A.1. Comenzando...

Lo primero que vemos al ejecutar la aplicación es la portada mientras se cargan los datos. Esto puede tardar unos pocos segundos. A continuación, nos encontramos con el menú principal que nos ofrece las siguientes opciones:

- **Comenzar relato**: Esta opción nos da acceso a la funcionalidad principal de la aplicación, donde se narrará la historia y el camino del usuario. Explicado más adelante en la sección A.2
- **Guardar coordenadas**: Esta opción nos ofrece la posibilidad de guardar las WiFi's con sus coordenadas. Explicado más adelante en la sección A.3
- **Crear código QR**: En esta opción podemos acceder a crear nuestros propios códigos para crear más narrativa. Explicado más adelante en la sección A.4
- **About**: En esta sección podemos ver información sobre los alumnos y los directores del presente proyecto.A.5

(a) Inicio (b) Menú principal

Figura A.1: Pantallas de inicio y menú principal

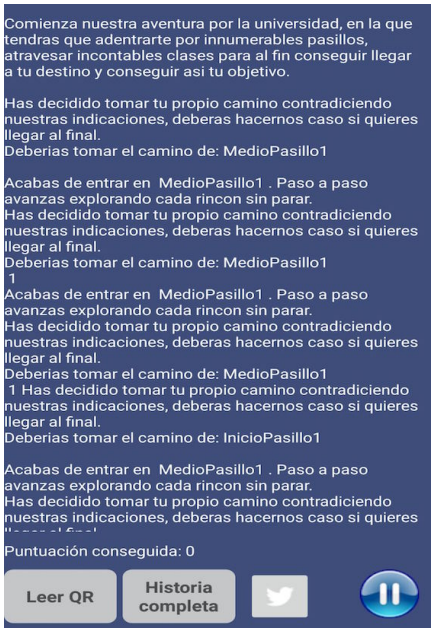
- **Valóranos**: En esta sección podremos puntuar la aplicación. Explicado más adelante en la sección A.5

A.2. Comenzar relato

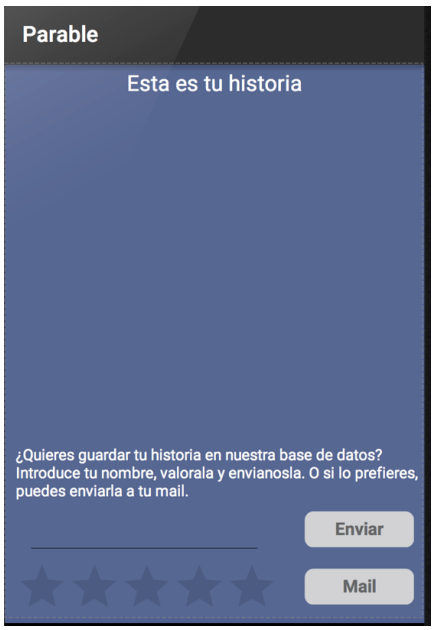
Una vez que el usuario accede a esta funcionalidad, la aplicación empezará a narrar su camino y dándole *feedback* de si está haciendo caso a la misión encomendada o no. Esta misión se podrá leer en esta misma sección de la aplicación y PARABLE le indicará el camino a seguir para completarla. En esta pantalla tendremos cuatro botones:

- Leer QR: Al pulsar este botón se activará la funcionalidad de leer un QR. El posicionamiento se interrumpirá mientras se esté ejecutando la lectura (ya que se da por hecho que el usuario permanecerá quieto) y una vez leído correctamente el código, la información que contiene se añadirá a la narrativa.

- Historia completa: Al pulsar este botón, la aplicación nos mostrará toda la historia completa junto a una valoración de la misma donde podremos darle un nombre y añadirla a la página web o enviarla por mail. Esta actividad puede ser vista en la imagen **??**.



(a) Ejemplo de narración de una historia



(b) Actividad que se muestra al pulsar Historia completa

Figura A.2: Funcionalidades principales de PARABLE

- Twitter: Al pulsar este botón se ejecutará la aplicación de terceros de twitter donde se podrá tuitear la ultima narración que nos ha ofrecido la aplicación.

- Play/Pause: Este botón pausa y reactiva la aplicación por si acaso no nos encontramos en disposición de continuar con la narración. Por ejemplo, salir de la zona mapeada deliberadamente para realizar otras actividades que nada tienen que ver con PARABLE.

A.3. Guardar coordenadas

Al lanzar esta opción se nos mostrarán la WiFi's que encuentra a su alrededor con sus intensidades, su dirección MAC y su nombre. Ver figura A.3a. Podremos pulsar en cualquiera de ellas y nos enviará a la siguiente pantalla donde introduciremos las coordenadas que creamos convenientes. Ver figura A.3b. Una vez que las coordenadas han sido correctamente introducidas en la base de datos se nos mostrará un mensaje temporal de la correcta ejecución.

(a) WiFi's encontradas a nuestro alrededor

(b) Pantalla para guardar coordenadas

Figura A.3: Pantallas de las redes y inserción de coordenadas

A.4. Crear código QR

Al lanzar esta opción tendremos la posibilidad de crear códigos a mayores para hacer nuestra narración más completa y enriquecerla. Solo tendremos que introducir el texto que queramos, por ejemplo: Has decidido entrar en el Aula 16. Luego deberemos imprimir, lógicamente, este código y colocarlo en el aula correspondiente. La aplicación, internamente, se ocupa de almacenar el código con una referencia única para que no se puedan falsificar.

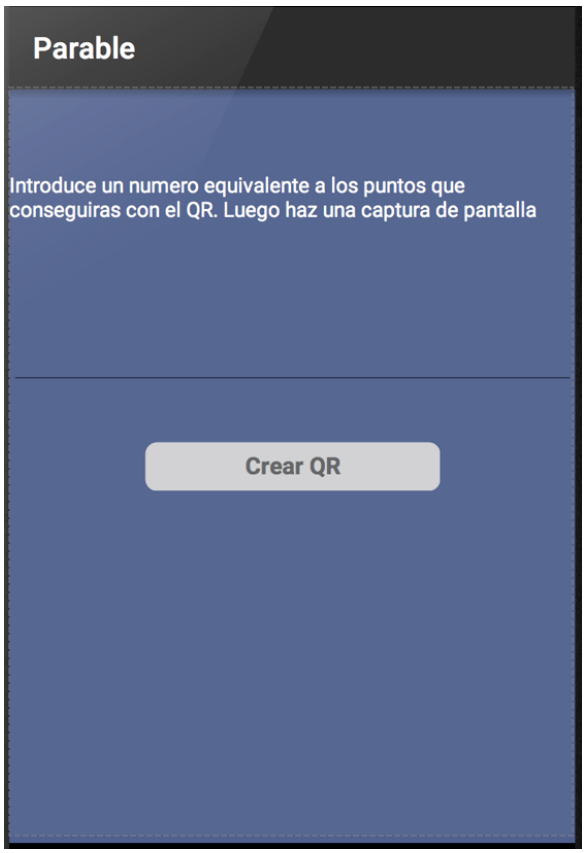


Figura A.4: Funcionalidad con la que podemos crear nuestros propios códigos

A.5. About y valóranos

AL pulsar en estas funcionalidades accederemos a información sobre los alumnos y los directores que han participado en PARABLE. Si se pulsa 10 veces sobre la pantalla que sale al pulsar el botón about encontrarás una sorpresa. En la sección de Valóranos, tendremos que introducir nuestro nombre, una reseña y una puntuación mediante estrellas que se guardará en nuestra base de datos para obtener el *feedback* de los usuarios.

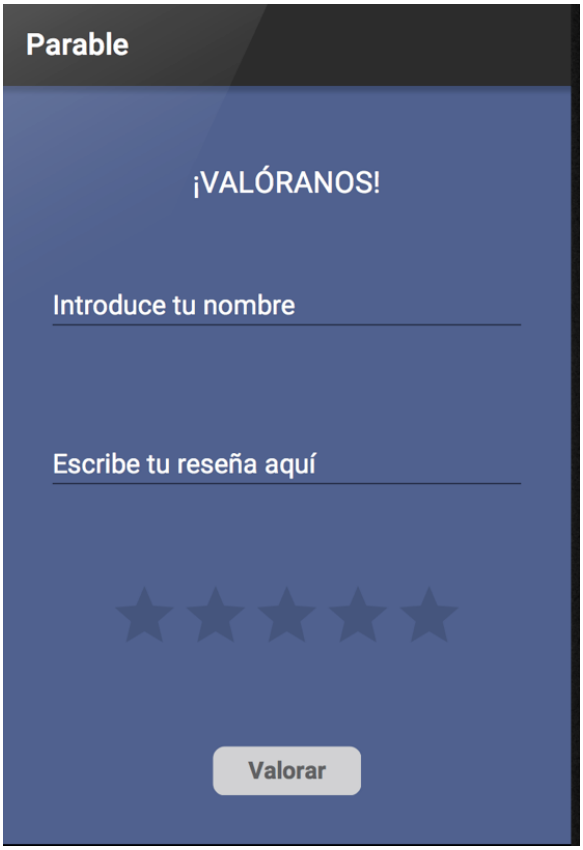


Figura A.5: Valoración de la aplicación por parte de los usuarios

Bibliografía

Podemos saber poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho que hacer.

Alan Turing

Android5.0. Android 5.0. 2014. Disponible en `https://www.android.com/intl/es_es/versions/lollipop-5-0/` (último acceso, Junio, 2016).

B, B. *A Spiral Model of Software Development and Enhancement*. IEEE Computer, 1988.

Bahl, P. y Padmanabhan, V. Radar: an in-building rf-based user location and tracking system. 2000.

Bekkelien, A. *Bluetooth Indoor Positioning*. Tesis Doctoral, University of Geneva, 2012.

Enrique López Mañas, J. P. H., Francisco Javier Moreno Nacarino. Avanti. 2010. Disponible en `http://eprints.ucm.es/11048/` (último acceso, Junio, 2010).

Galactic. The stanley parable. Disponible en `http://www.stanleyparable.com/`.

Gelbukh, A. y Bolshakov, I. Avances y perspectivas de procesamiento automático de lenguaje natural. Disponible en `http://www.gelbukh.com/CV/Publications/2000/IPN-Proc-Leng-Nat.htm`.

Mark Weiser, B., Gold. Ubiquitous computing. Disponible en `http://www.research.ibm.com/journal/sj/384/weiser.html`.

Nogee, A. Ready or not, mobile location technology is here! 2001.

Pressman, R. S. *Ingeniería del Software*. McGraw-Hill Interamericana, 2005.

RESEARCH, A. Location based services:analysis of carrier spending, subscribers, devices and applications for handset-based and telematics services. 2004.

SAHA, M. A., D. *Pervasive Computing: A Paradigm for the 21st Century*, vol. 36. IEEE Computer, 2003.

SIMPLEXT. Disponible en `http://www.simplext.es/`.

WEISER, M. Mark weiser. Disponible en `https://es.wikipedia.org/wiki/Mark_Weiser`.

WEISER, M. Ubiquitous computing. Disponible en `http://www.ubiq.com/hypertext/weiser/UbiHome.html`.

WEISER, M. The computer for the 21st century. Disponible en `https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf`.

WEISER, M. *The Computer for the 21st century.*, vol. 256. IEEE Computer, 1991b.

WEISER, M. Ubiquitous computing. Disponible en `http://www.ubiq.com/hypertext/weiser/UbiHome.html`.

WIKIPEDIA. Computación ubicua. Disponible en `https://es.wikipedia.org/wiki/Computación_ubicua`.

WIKIPEDIA. Cuarta pared. Disponible en `https://es.wikipedia.org/wiki/Cuarta_pared`.

WIKIPEDIA. Procesamiento de lenguaje natural. Disponible en `https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales`.

WIKIPEDIA. siri. Disponible en `https://es.wikipedia.org/wiki/Siri`.

WIKIPEDIA. Rfid. Disponible en `https://es.wikipedia.org/wiki/RFID`.

WIKIPEDIA. Beacon. Disponible en `https://es.wikipedia.org/wiki/Beacon`.